SEE
FOLLOWING
NOTE

VOIR
NOTE
CI-APRÈS

PARTS OF THE ORIGINAL DOCUMENT

ARE OF POOR REPRODUCTIVE QUALITY

CERTAINES PARTIES DU DOCUMENT ORIGINAL

SONT DE PIÈTRE QUALITÉ DE REPRODUCTION

THE FOLLOWING PAGES WERE NOT
RECEIVED WITH ORIGINAL DOCUMENT AND
THEREFORE HAVE NOT BEEN REPRODUCED:

APP. A6 TUES 18 JUN PP. 3-4

PUISQUE LES PAGES SUIVANTES N'ONT PAS ÉTÉ
REÇUES AVEC LE DOCUMENT ORIGINAL
ELLES N'ONT PAS ÉTÉ REPRODUITES:

TP 8328 E

SOFTWARE DEVELOPMENT FOR A MICROPROCESSOR BASED CONTROLLER FOR AC-DC-AC DIESEL-ELECTRIC FREIGHT LOCOMOTIVE POWER UNIT CONTROL

Prepared for:

Transportation
Development Centre

Transport Canada

by

BBC Brown Boveri Canada Inc.

February 1987

CA1
DTC 800
87
S52
c.1

89-00634/

TP 8328 E

SOFTWARE DEVELOPMENT FOR A MICROPROCESSOR BASED CONTROLLER FOR AN AC-DC-AC DIESEL-ELECTRIC FREIGHT LOCOMOTIVE POWER UNIT CONTROL

Library and Information Centre /
Centre de Documentation (GSLA)
RECEIVED - REÇU LE
NOV 19 1987
Transport Canada / Transports Canada

Prepared for:

Transportation
Development Centre

Transport Canada

by

Dr. R. Bonert, University of Toronto

M.Nixon, Synitek

for

BBC Brown Boveri Canada Inc.

February 1987

365452

The contents of this report reflect the views of BBC Brown Boveri Canada Inc. and the University of Toronto, and not necessarily the official view or opinions of the Transportation Development Centre of Transport Canada.

Un sommaire en français de ce repport est inclus avant la Table des matières.

Project Team

Dr. R. Bonert, University of Toronto
Dr. S. B. Dewan, " " "
R. Knox, " " "
P. Bubonja, " " "

J. van den Bosch, BBC Brown Boveri Canada Inc.
R. Bhatia, " " " " "
M. Moreau, " " " " "
R. Schyle, " " " " "

M. Nixon, Synitek Inc.

Transport Canada Transports Canada

Development Développement

REPORT DOCUMENTATION FORM

1. Transport Canada Report No.
TP 8328E

2. TDC Project No.
4560-6753

3. Recipient's Catalogue No.

4. Title and Subtitle
Software Development for a Microprocessor Based
Controller for an AC-DC-AC Diesel-Electric Freight
Locomotive Power Unit Control

5. Report Date
February 1987

6. Performing Organization Report No.

7. Author(s)
Dr. R. Ponert, M. Nixon

8. Transport Canada File No.
D1465-371

9. Performing Organization Name and Address
BBC Brown Boveri Canada Inc.
4000 Trans-Canada Highway,
Pointe Claire, Quebec.
Canada
H9R 1B2

10. DSS File No.
11 SD.T8200-1-1563

11. DSS or Transport Canada Contract No.
OSD 82-00058

12. Sponsoring Agency Name and Address
Transportation Development Centre (TDC)
200 Dorchester Blvd. West,
Suite 601, West Tower,
Montreal, Quebec
H2Z 1X4

13. Type of Report and Period Covered
Final Report

14. Sponsoring Agency Code

15. Supplementary Notes

16. TDC Project Officer
C.A. Versailles

17. Abstract

The objective of the project was the development of advanced microprocessor software for AC-DC-AC locomotive control.

The first task of the project was the development of a real time interactive simulation and test (RIST) system. The RIST system creates a real time model of the locomotive's inputs and dynamic system responses, (engine, transmission and wheel respoonse to control signals) when connected to the MICAS control computer.

The second task was the development of MICAS control software. This work addressed three major areas :traction control, monitoring/diagnostics and diesel engine control. The work in these areas developed not only optimized control algorithms, but also explored control strategies unique to the AC-DC-AC transmission system. The work on monitoring produced both software and guidelines for future hardware development. The diesel engine control software is designed to improve efficiency by optimizing engine operating point to load demand. Also explored are the potential fuel savings through computerized multiple unit control.

The third task was the establishment of a software/hardware integration test facility. This facility, comprising the MICAS computer, inverter control, inverter, motor and power supply, permits safe testing of new software in a controlled, low-power environment.

18. Key Words
Microprocessor Control
AC-DC-AC Diesel-Electric
Locomotive

19. Distribution Statement
Limited number of copies available from the Transportation Development Centre on request

20. Security Classification (of this report)
Unclassified

21. Security Classification (of this page)
Unclassified

22. No. of Pages
343

23. Price

TDC/CDT 79-005
Rev 86

Canada

Transports Canada Transport Canada

Développement Development

FORMULE DOCUMENTAIRE DU RAPPORT

1. Nº du rapport Transports Canada
TP 8328E

2. Nº de l'étude — CDT
4560-6753

3. Nº de catalogue du destinataire

4. Titre et sous-titre
Software Development for a Microprocessor Based
Controller for an AC-DC-AC Diesel-Electric Freight
Locomotive Power Unit Control

5. Date du rapport
Février 1987

6. Nº du rapport de l'organisme exécutant

7. Auteur(s)
Dr. R. Bonert, M. Nixon

8. Nº de dossier Transports Canada
D1465 - 371

9. Nom et adresse de l'organisme exécutant
BBC Brown Boveri Canada Inc.
4000 Ch. Transcanadienne
Pointe Claire, Québec
H9R 1B2

10. Nº de dossier — ASC
11SD T8200-1-1563

11. Nº de contrat — ASC ou Transports Canada
OSD 82-00058

12. Nom et adresse de l'organisme parrain
Centre de développement des Transports,
200 ouest boul. Dorchester,
Suite 601, Tour Ouest
Montreal, Quebec.
H2Z 1X4

13. Genre de rapport et période visée
Rapport Final

14. Code de l'organisme parrain

15. Remarques additionnelles

16. Agent de projet — CDT
C.A. Versailles

17. Résumé

Le projet avait pour objectif de dévolopper un logiciel avancé de microprocesseur pour le contrôle de locomotive c.a.-c.c -c.a.

La première tâche du projet était de développer un système de simulation d'interaction et d'essai en direct ("real time interactive simulation and test system") (RIST). Le système RIST crée un modèle en temps réel des entrées (données) de la locomotive et des réactions du système dynamique,(réaction du moteur, de la transmission et des roues aux signaux de contrôle) lorsque branché à l'ordinateur de contrôle MICAS.

La deuxième tâche était le développement de logiciel de contrôle MICAS. Ce travail portait sur trois secteurs majeurs: le contrôle de traction, supervision/diagnostics et commande de moteur Diesel. Le travail dans ces secteurs a non seulement développé des algorythmes de contrôle optimisés, mais a aussi exploré les stratégies de contrôle unique du système de transmission c.a.-c.c.-c.a.. Le travail sur le contrôle a produit le logiciel et des directives pour le développement futur de matériel. Le logiciel de contrôle du moteur Diesel est conçu pour améliorer le rendement en optimisant le point de fonctionnement du moteur selon la demande de charge. L'économie possible de carburant par un contrôle d'unités multiples informatisé a aussi été exploré.

La troisième tâche était l'établissement d'une installation d'essai d'intégration du logiciel et du matériel. Cette installation, composée de l'ordinateur MICAS, de la commande d'inverseur, de l'inverseur, du moteur et du bloc d'alimentation, permet le test sécuritaire de logiciel nouveau dans un milieu contrôlé de basse puissance.

18. Mots-clés
Développement de logiciel pour régulateur
en technique de microprocesseurs pour locomotives
Diesel-électriques c.a.-c.c.-c.a. à marchandises

19. Diffusion
Des copies en quantité limitées
sont disponibles au Centre de développement des transports.

20. Classification de sécurité (de ce rapport)
Non classifié

21. Classification de sécurité (de cette page)
Non classifié

22. Nombre de pages
343

23. Prix

CDT/TDC 79-005
Rev 86

Canada

Acknowledgement

This report, with the exception of Section 9, was prepared by:

Dr. R. Bonert
Department of Electrical Engineering
University of Toronto
Toronto, Ontario

BBC Brown Boveri Canada Inc. wishes to acknowledge and thank Dr. R. Bonert and his Project Team from the University of Toronto for their major role in this project.

BBC also wishes to thank Mr. M. Nixon for his valuable assistance in completing the project and Section 9 of this report.

SOFTWARE DEVELOPMENT FOR A MICROPROCESSOR BASED CONTROLLER FOR AN AC-DC-AC DIESEL-ELECTRIC FREIGHT LOCOMOTIVE POWER CONTROL UNIT

SUMMARY

The objective of the project was the development of advanced microprocessor software for AC-DC-AC locomotive control using the MICAS computer. The computer is in service on an experimental AC-DC-AC locomotive using a software program that emulates the conventional analog control.

The program was established on the basis that the development and testing of the software could not utilize the locomotive for protracted time periods. In order to meet the objective, a three step program was developed. These steps were developing a simulated locomotive environment for test purposes, software for the locomotive controller and developing a software/hardware integration test facility.

The first step involved the development of a real time interactive simulation and test (RIST) system and support system for programming. The functions of the RIST system are to interface with the MICAS computer and permit locomotive control inputs, simulate in real time the dynamic responses of the locomotive and permit data collection and monitoring of the test software's performance. The system comprises a VME68000 computer which operates the locomotive model program and handles the control and MICAS interfaces. A 6809 single board computer performed the data acquisition and display functions.

In addition to the RIST hardware, a comprehensive modular real time locomotive simulation software package was developed. This software package permits simulation of the engine governor response, the AC-DC-AC transmission and the locomotive wheel/rail interface condition variations. An additional software function is the data acquisition and display function which permits complete monitoring of the effect of the MICAS test software on the locomotive model software.

The second step was the development and testing of MICAS control software for traction control, monitoring and diagnostics and diesel engine control.

The traction control software development explored various approaches to both the detection and the correction of the wheelslip condition. In addition, it explored new variations in configuring the AC-DC-AC drive control to improve adhesion performance.

The monitoring and diagnostic software development explored the use of MICAS for data acquisition, storage, processing, storage of processed data and document provision or display of the processed data. Several routines were developed which permit the MICAS system to acquire and communicate system data. In addition, hardware guidelines for future systems were developed.

The diesel engine control software development optimized engine fuel efficiency by optimizing the engine operating point to the power demand. Additionally, it explored the potential for fuel savings through the use of computerized multiple unit control.

The third step in the program was the establishment of a laboratory facility to permit testing of the developed software on real locomotive hardware. The laboratory consists of a MICAS computer, inverter control, inverter, power supply, traction motor and control console. This facility permits testing of the software in a low power controlled environment where the software can be exercised and unforeseen reactions safely observed and corrected.

SOMMAIRE

Le projet avait pour objectif de développer un logiciel avancé de microprocesseur pour locomotive c.a.-c.c.-c.a., en se servant de l'ordinateur MICAS. L'ordinateur est en service sur une locomotive expérimentale c.a.-c.c.-c.a. et utilise un programme de logiciel qui simule le contrôle analogue conventionnel.

Le progamme a été établi sur la base que le développement et les essais du logiciel ne pouraient pas se servir de la locomotive durant des périodes de temps prolongées. Afin de réaliser l'objectif établi, un programme à trois étapes a été développé. Voici les étapes: le développement d'un milieu de locomotive simulé pour fins d'essais,du logiciel pour le contrôleur de locomotive et le développement d'une installation d'essai d'intégration du logicïel et du matériel.

La première étape comprenait le développement d'un système de simulation d'interaction et d'essai en direct ("real time interactive simulation and test system") (RIST) et d'un système de support pour la programmation. Les fonctions du système RIST sont de servir d'interface entre l'ordinateur MICAS et les entrées provenant de la simulation de la locomotive, de simuler en temps réel les réponses dynamiques de la locomotive et de permettre l'acquisition de données et le contrôle de la performance du logiciel d'essai. Le système comprend un ordinateur VME68000 qui opère le programme de modèle de locomotive et qui manoeuvre le contrôle et les interfaces de MICAS. Un ordinateur 6809 a effectué les fonctions d'acquisition et d'affichage de données.

En plus du matériel RIST, un ensemble modulaire de logiciel de simulation en temps réel de locomotive a été développé. Cet ensemble de logiciel permet la simulation de la réaction du régulateur du moteur, de la transmission c.a.-c.c.-c.a. et des variations de la condition provenant de l'interaction roue/rail entre la locomotive et le chemin de fer. Une fonction additionnelle du logiciel est l'acquisition et l'affichage des données qui permet la supervision complète de l'effet du logiciel d'essai MICAS sur le logiciel de modèle de locomotive.

La deuxième étape était le développement et l'essai du logiciel de contrôle MICAS pour le contrôle de la traction, la supervision et les diagnostics et le contrôle du moteur Diesel.

Le développement du logiciel de contrôle de traction a exploré des différentes méthodes pour détecter et corriger le glissement des roues ("wheelslip"). De plus, il a examiné de nouvelles variations dans la configuration de la commande d'entrainement c.a.-c.c. -c.a pour améliorer la performance d'adhésion.

Le développement de logiciel de supervision et de diagnostic a exploré l'usage du MICAS pour l'acquisition de données, la mise en mémoire, le traitement, la mise en mémoire des données traitées et la possibilité de fournir des documents ou d'afficher les données traitées. Plusieurs programmes ont été dévelopés qui permettent au système MICAS d'acquérir et de communiquer les données du système. De plus, des directires de matériel ont été elaborées pour des systèmes futurs.

Le développement de logiciel de contrôle du moteur Diesel a optimisé le rendement du carburant du moteur en optimisant le point de fonctionnement du moteur selon la demande de puissance. De plus, il a étudié la possibilité d'economies en carburant par l'usage d'un contrôle d'unités multiples informatisé.

La troisième étape du programme était d'établir une installation de laboratoire permettant de tester le logiciel développé sur le matériel actuel de la locomotive. Cette installation comprend un ordinateur MICAS, un système de commande d'inverseur, l'inverseur, un bloc d'alimentation, un moteur de traction et un pupitre de contrôle. Cet aménagement permet de tester le logiciel dans un milieu contrôlé de basse puissance où le logiciel pruy être appliqué et des réactions imprévues peuvent être observées sans risque et être corrigées.

Table of Contents

SOFTWARE DEVELOPMENT FOR A MICROPROCESSOR-BASED CONTROLLER FOR AC-DC-AC DIESEL ELECTRIC FREIGHT LOCOMOTIVE POWER CONTROL UNIT

		Page
1	Introduction	1
2	**Real Time Interactive Simulation and Test system (RIST) and support systems for programming**	3
2.1	Introduction	3
2.2	Necessity and definition of RIST	3
2.2.1	Design and building of a controller	3
2.2.2	Necessity of an integrated support system to develop locomotive controls	4
2.2.3	General definition of the RIST system	4
2.3	The RIST system for the MICAS microprocessor control system	5
2.3.1	The MICAS microprocessor control system and its interface	5
2.3.2	The hardware of the RIST system	7
2.3.3	The software of the RIST system	9
2.3.4	The program module IOLOCO	10
2.3.5	The program DATDIS2	10
2.3.6	The program module MADB, MDABL	13
2.4	Support systems for programming	13
2.4.1	Support for the VME computer	13
2.4.2	Support for the single board computer 6809	15
2.4.3	Support for the MICAS computer	15

Page

3 **Locomotive System Simulation** 17

3.1 Introduction 17

3.2 Simulation of the diesel engine operation 17

3.2.1 System description and modelling of the diesel engine and its controls 17

3.2.2 Quantities of the diesel engine model 22

3.2.3 Program DLOCO and examples of the diesel engine simulation 26

3.3 Simulation of the electric drive 29

3.3.1 System description and modelling of the electric drive and its controls 29

3.3.2 Quantities of the electric drive model 32

3.3.3 The program ELOCO and examples of the torque command control system 35

3.4 Simulation of the traction and the train 37

3.4.1 System description and simulation of the traction and the train 37

3.4.2 Quantities of the train and traction simulation 41

3.4.3 The programs MLOCO, ASMCURVE, CURVEGEN, TLOCO and examples of the traction and train simulation 44

3.5 Real time simulation with the LOCO 47

3.6 The complete RIST system with the MICAS microprocessor control system 49

4 **System Control Studies** 51

5 **Traction Control** 53

5.1 System description and analysis 53

Page

5.1.1 Traction force characteristic 53

5.1.2 The ac drive 53

5.1.3 Sensing of the slip condition 55

5.2 Specifications of the traction force control system 56

5.3 Control strategy 57

5.4 Search strategy for optimal utilization of the friction 60

5.5 Simulation of the proposed microslip control 63

5.5.1 Parameter selection for the microslip control study 64

5.5.2 Simulation results 65

5.6 Implementation of the proposed microslip control 72

5.6.1 Torque feedback 72

5.6.2 Implementation of the microslip control with stator current feedback 73

5.6.3 Locomotive speed measurement 73

5.6.4 Implementation of the control on the MICAS system 74

5.7 The old MICAS slip limit control with rate of change sensing 77

5.8 Comparison to other potential slip controls with different drives 82

5.8.1 General influence of the drive characteristic 82

5.8.2 Friction utilization potential of different drives 85

6 Diesel Engine Overload Protection and Maximum Power Utilization 87

6.1 Concept of the overload control 87

6.2 Simulation of an overload protection control 87

Page

6.3 Maximum diesel power utilization with the load factor control 90

7 **Monitoring and Diagnostics** 92

7.1 Tasks and capabilities of the MICAS system 92

7.2 Strategy to use MICAS in a data acquisition system 93

7.3 Concept of a data acquisition system with MICAS and a PC 94

7.3.1 General capability 94

7.3.2 Data link 94

7.3.3 PC properties and requirements 94

7.3.4 MICAS system properties and requirements 95

7.3.5 Principle function 95

7.4 Verification of the MICAS PC link for data acquisition 96

7.4.1 Timing structure, communication and processing capacity 96

7.4.2 MICAS system loading analysis 98

7.5 The MICAS communications operating system 102

7.5.1 The program block AUX1 (CIONIT) and AUX2 (CIOBGR) 102

7.5.2 The program block SERE (CIOINT_V5) 103

7.5.3 Program CODA (CIOSYS) 103

7.5.4 The program FADA (CIOFDS) 103

7.5.5 The PDP11 program 104

7.6 Results 105

8 **Diesel and Locomotive Speed Controls and Their Impact on Fuel Usage** 107

8.1 Potential for fuel savings by controls 107

Page

8.2 Fuel efficiency control for multiple traction with speed control 108

8.3 Theoretical fuel savings 109

8.4 Fuel saving by train control 112

9 Laboratory Test by BBC Using an AC Traction Motor and an Inverter 114

9.1 Introduction 114

9.2 Equipment 114

9.3 Test Results 118

9.3.1 Configuration 118

9.3.2 Power Supply Test 118

9.3.3 Commutation Test 119

9.3.4 Motor Test 120

10 Conclusions 134

References 136

Appendix A1 - Keyboard Commands

Appendix A2 - Module Map LOCO, Procedure Descriptions and Real-time VME Resident Software

Appendix A3 - Display Computer Software

Appendix A4 - Support Program Listings

Appendix A5 - MICAS Microslip Program

Appendix A6 - MICAS-PDP11 Communications System

LIST OF FIGURES

Fig. No.	Title	Page
2.1	RIST System Structure	5
2.2	MICAS/RIST General Interface Structure	6
2.3	MICAS/RIST Interface with Input and Output Values	7
2.4	RIST System Hardware	8
2.5	Display with Three Variables	11
2.6	DATDIS2 Program Structure	12
2.7	Connection Between Computer Systems	14
3.1	Diesel Engine Operation Block Diagram	18
3.2	Diesel Engine Basic Control Structure	19
3.3	Diesel Engine Control Structure	20
3.4	Turbo-Charger Control Structure	21
3.5	Diesel Engine Model Digram	22
3.6	Engine Simulation Data	24
3.7	Engine Simulation Model Block Diagram	25
3.8	Example of Diesel Simulation Variables	27
3.9	Example of Diesel Simulation Variables	28
3.10	Drive Simulation Diagram	30
3.11	Inverter Simulation Block Diagram	30
3.12	Traction Simulation Block Diagram	31
3.13	Traction Diagram	33
3.14	Drive Simulation Block Diagram	34
3.15	Torque Command Simulation, Notch 7	36
3.16	Torque Command Simulation, Notch 8	36

Fig. No.	Title	Page
3.17	Torque Command Simulation, Intermediate	37
3.18a	Illustration of Characteristic Changes	38
3.18b	Illustration of Characteristic Changes	38
3.18c	Illustration of Characteristic Changes	39
3.19	Traction & Train Simulation Diagram	40
3.20	Closed Loop Transfer Function	41
3.21	Traction Force/Slip Characteristics - First Type	43
3.22	Traction Force/Slip Characteristics - Second Type	43
3.23	Example of Slip Event	44
3.24	Example of Slipping Transient	45
3.25	Example of Slipping Transient	46
3.26	LOCO Program Structure	48
3.27	Complete RIST and MICAS System	50
5.1	Four-Motor Drive Structure	54
5.2	Tractive Force Control Specifications	56
5.3	Microslip Control with Locomotive Model	58
5.4	Microslip Control Example	59
5.5	Drive Characteristic with Maximum Microslip Control	60
5.6	Search Steps	61
5.7	Friction Utilization Control Algorithm	62
5.8	Microslip Control Block Diagram	63
5.9	Microslip Control - Traction Characteristics	64
5.10	Microslip Simulation Example	66

Fig. No.	Title	Page
5.11	Slip Limitation to Set Values	67
5.12	Control of Changes in Slip Conditions	67
5.13	Microslip Control - Backward Search	68
5.14	Microslip Control - Backward Search	69
5.15	Microslip Control - without Maximum Search	70
5.16	Microslip Control - with Maximum Search	70
5.17	Microslip Control - without Maximum Search and with Limited Slip	71
5.18	MICAS Program Blocks and Connections	75
5.19	MICAS Program Blocks and Connections	76
5.20	Slip Control with Rate-of-Change Sensing	77
5.21	Slip Control with Rate-of-Change Sensing	78
5.22	Slip Control with Rate-of-Change Sensing	78
5.23	Slip Control with Rate-of-Change Sensing	79
5.24	Slip Control with Rate-of-Change Sensing	80
5.25	Slip Control with Rate-of-Change Sensing	81
5.26	Slip Control with Rate-of-Change Sensing	81
5.27	Drive Characteristic - Single Axle	82
5.28	Drive Characteristic - Multiple Axles	82
5.29	Drive Characteristic - Common Bus Control	83
5.30	Drive Characteristic - Common Bus Control	83
5.31	Effect of Steep Motor Characteristic	84
5.32	DC and AC Drive Characteristics	85
6.1	Overload Protection Control Simulation	88

Fig. No.	Title	Page
6.2	Overload Protection Control Example	89
6.3	Overload Protection Control Example	90
7.1	Sampling and Sending Timing Diagram	96
8.1	Fuel Efficiency Control	107
8.2	Fuel Efficiency Assumptions Graph	109
8.3	Fuel Usage vs Power	110
8.4	Fuel Usage - Multiple Traction	111
8.5	Fuel Saving vs Required Total Power	112
9.1a - 9.1e	Equipment Photos	121-127
9.2	Configuration with Motor	128
9.3	Configuration with Resistive Load	129
9.4	Power Supply Graph	130
9.5	Commutation Power Graph	131
9.6	Commutation Characteristic Graph	132
9.7	Motor Test Graph	133

LIST OF TABLES

3.1	Diesel Engine Data for Simulation	23
3.2	Traction Diagram Data	32
3.4	Parameterss Traction Force Characteristirs	42

1. Introduction

SOFTWARE DEVELOPMENT FOR A MICROPROCESSOR BASED CONTROLLER FOR AC-DC-AC DIESEL-ELECTRIC FREIGHT LOCOMOTIVE POWER CONTROL UNIT

Sections 2 through 8 of this report cover the software development part of the project.

This part of the project has two objectives.

The first is to develop a systematic approach and a support system, that allows efficient design and testing of software and hardware for microprocessor based controllers. The system should provide the engineer with efficient tools for the design and testing of microprocessor control software. It should accomodate the specific application of microprocessors for locomotive controls, and be general enough to support future developments in the microprocessor control of freight locomotives.

The second is to study and design control algorithms and develop software for specified freight locomotive controls.

To understand the first objective it is necessary to recall the many activities required to produce control software. A system analysis of the plant to be controled is required. For larger systems, the resulting model has to be verified by a simulation. The control has to be designed and its function is verified by a simulation at the model. The hardware is selected and the software has to be designed. The software has to be tested, first the software itself and then in conjunction with the hardware. Finally, the controller has to be tested with the plant. These tests can start with tests against simulted plant functions, as direct testing is too dangerous or uneconomical. Finally the controls are commissioned and the final test with the plant confirms the function of the system.

The testing is a very important part of the whole proces and often the most costly. Particularly in the case of a locomotive, which is in the field and offers few possibilities to set up a good test facility on board, a test system avoiding too much work at the plant becomes crucial.

In this project, a Real time Interactive Simulation and Test system (RIST) has been developed that supports simulation, software development, extensive hardware and software testing in the laboratory or factory and training of the commissioning engineer. This system will be described in the first part of the report.

The RIST system is used to analyze the locomotive system and to develop and test software for specific locomotive controls.

This is described in the second part of the report. The overload protection for the diesel and the dynamics of the wheel-rail contact will be discussed after a general analysis of the locomotive system has been made. Specific control problems as specified in the requirement definitions will be described afterwards. These are the microslip control, the locomotive speed control and the use of microprocessors for data acquisition and monitoring on the locomotive for performance control and maintenance. Together with the speed control, the impact of this control on fuel consumption in multiple traction will be studied.

LABORATORY TEST USING AN AC TRACTION MOTOR AND AN INVERTER

The test setup described in section 9 allows testing of the control software in a safe, controlled low power environment, while using the actual locomotive electrical equipment.

2. Real Time Interactive Simulation and Test system (RIST) and support systems for programming

2.1 Introduction

The simulation and test system and other support systems used for the project to develop and test microprocessor software are described in this section. This includes the RIST system and the general support systems for program development.

A general justification and specification of the RIST system will be given first. Then the microprocessor control system MICAS is shortly described to define the interface the RIST system has to accommodate.

After these general specifications the RIST system developed and built for this project is described. The description includes the hardware and the general software to support the operation of the RIST system.

At the end of this section other support systems necessary to operate the MICAS and RIST systems and to develop software for both systems are described.

2.2 Necessity and definition of RIST

To show the necessity of an integrated and efficient support system for the design and test of controllers the task of a controller design is described first. The Real time Interactive Simulation and Test system is then defined with reference to the required design tasks.

2.2.1 Design and building of a controller

To design a control a thorough analysis of the plant is necessary. The result is a model describing the function of the plant in terms of its output signals with respect to its input control signals. For more complex systems this model is verified by a computer simulation on a general computer. Often such a simulation is used to fine tune the model to give it exactly the known behavior of the plant. The simulation on the general computer is done in a general programming language and is normally not in real time.

After the analysis the control algorithms are designed. To verify the control strategy and the design, the controllers are simulated with the system, to see whether the expected performance is achieved.

After the verification of the control algorithms the controller is built. In the case of a microprocessor based control system this means often selecting an already built hardware system or taking the one the company usually uses and writing the software, which implements the desired control. Writing the software is usually constrained by the operating system and the programming language, which comes with the selected hardware.

After the controller has been built, it has to be tested. The tests have two parts, the pretests in the factory and the final tests with the plant. Input signals are applied for testing and by observing the output signals it is judged whether the controller functions as desired. In complex systems this kind of testing does not allow a final judgement whether all the complex control tasks will be performed as desired. Only the basic functions of the controller can be verified. Sometimes very elaborate test systems are built to reach a full testing. These systems come close to being a real time system simulator, if they

really allow a thorough testing.
Finally the controller is commissioned and the final tests of the controller connected to the plant are done. These final tests can be very difficult to carry out, as the environment is hostile to testing or the risks are very high if a test fails and the danger of damaging the plant occurs.

2.2.2 Necessity of an integrated support system to develop locomotive controls

The control of a locomotive is a complex task. Therefore a system simulation is required to analyze the system thoroughly enough to develop controllers for it. The control strategies have to be tested against the model. Finally the microprocessor controller with its software has to be tested. To allow more than basic function tests a test facility is required, that provides locomotive-like signals for a complete test of the designed software. This complex test facility will be based on a computer.
A thorough and complete function test is a must, as the locomotive is an absolute hostile environment for final tests. There is little room on a locomotive to set up appropriate test facilities. It is a considerable effort to operate a locomotive such, that certain conditions of operation occur. But such reproducible conditions are necessary to test and adjust the designed controllers. Furthermore a locomotive under real test conditions is very often in a remote location far away from any technical support.
Another important advantage of a full function test in the factory is the possibility to prepare the engineer for the commissioning. In this case all the specialists who designed the system can be reached easily if major problems occur during the system tests.
Using a computer to simulate the system and using another one to set up an appropriate test environment suggest integrating these two tasks into one computer system.

2.2.3 General definition of the RIST system

The Real time Interactive Simulation and Test system is in principle a process computer like the microprocessor based controller. It differs from a general computer as it has a process interface for the input and output of electrical signals and it operates in real time. Its general structure is shown in fig. 2.1.
The communication I/O contains all kinds of interfaces to allow one to program the system and to observe its operation. It should include the possibility of accessing most system variables of the simulation and display them in real time on an oscilloscope, chart recorder or video screen.
The process I/O is the interface to the microprocessor controller. The electrical signals at this interface are identical to the digital and analog signals the microprocessor controller will see when it is connected to the real plant. Therefore the controller sees no difference between being connected to the plant or the RIST system.
The operator I/O contains the control devices set by the operator of the plant. They may be switches, potentiometers or other command devices used by the plant operator to control the process. This interface allows a realistic interactive control of the plant in the way the real system would be controlled.

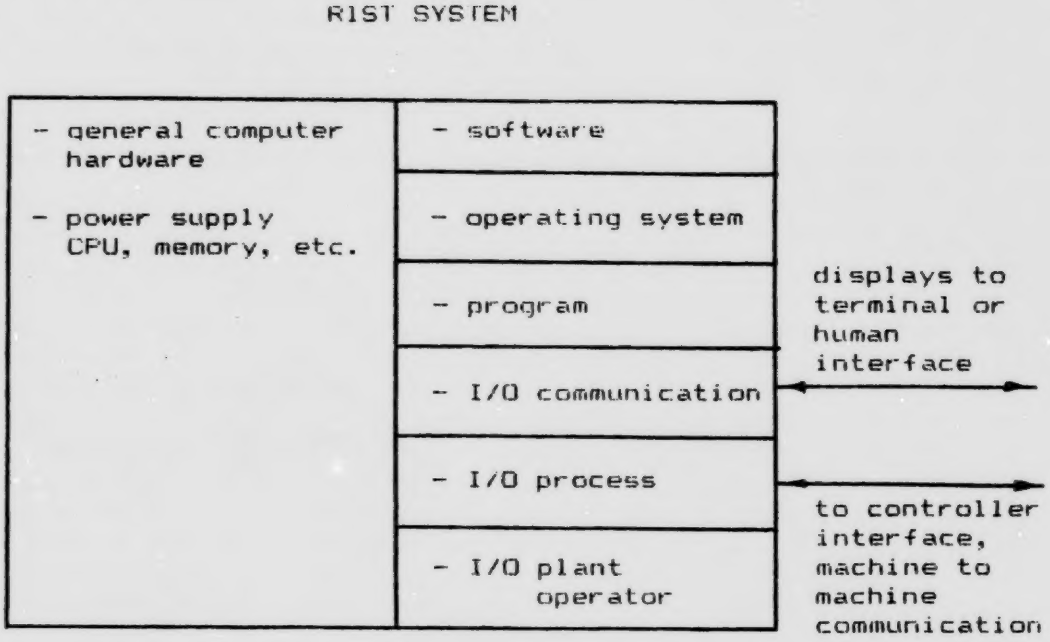

RIST SYSTEM
- general computer hardware
- power supply CPU, memory, etc.
- software
- operating system
- program
- I/O communication
- I/O process
- I/O plant operator
displays to terminal or human interface
to controller interface, machine to machine communication

Fig. 2.1 RIST System Structure

The program runs the RIST system such that the process interface looks to the microprocessor controller like the later plant connection. That means the program contains the general process model, models of the sensors providing the signals for the microprocessor controller and models of the input actuators controlled by the microprocessor control signals. Beyond this the signals are of identical voltage or current as specified for the microprocessor controller interface.

If the RIST system is set up like this it allows a complete system simulation and analysis. It provides complete hardware and software tests of the system control including transients from operation and fault conditions. The identical interface signals have the advantage, that no program changes are required after the factory tests to adapt to the real plant. Furthermore this interface allows an excellent preparation of the commissioning as the signals at the microprocessor controller input and output are identical or very similar to what will be observed at the real plant. The realistic operator interface allows ahead familiarization of the commissioning personnel with the plant's controls.

2.3 The RIST system for the MICAS microprocessor control system

2.3.1 The MICAS microprocessor control system and its interface

The MICAS microprocessor control system has in principle the same structure as shown in fig.2.1 for the RIST system except the operator I/O. It is a

system of printed circuit boards which are inserted in a card rack. The printed circuit boards contain the microprocessor based central processing unit, memory cards, timing and interrupt controllers and input output cards.
Among the I/O cards is a serial port card, which serves as an interface to a terminal to provide human access to the MICAS microprocessor control system. The other I/O cards provide the process interface. They include digital input and output lines and analog inputs and outputs. All these signals have a voltage level of 15 V.
The process interface signals from the microprocessor control system are then connected to other cards containing a general electronic process interface. This general electronic interface mainly provides electric isolation from the plant and signal level adaptation. Sometimes the general electronic interface provides some filtering.

For the software development and testing the microprocessor control system interface at the 15 V level is the appropriate interface for the RIST system. The general electric interface changes mainly the electrical properties of the signals and is of process specific nature. If the signals are influenced by the general electric interface i.e. by filtering, this is considered to be part of the plant property and will be included into the plant model. The general structure is shown in fig. 2.2.

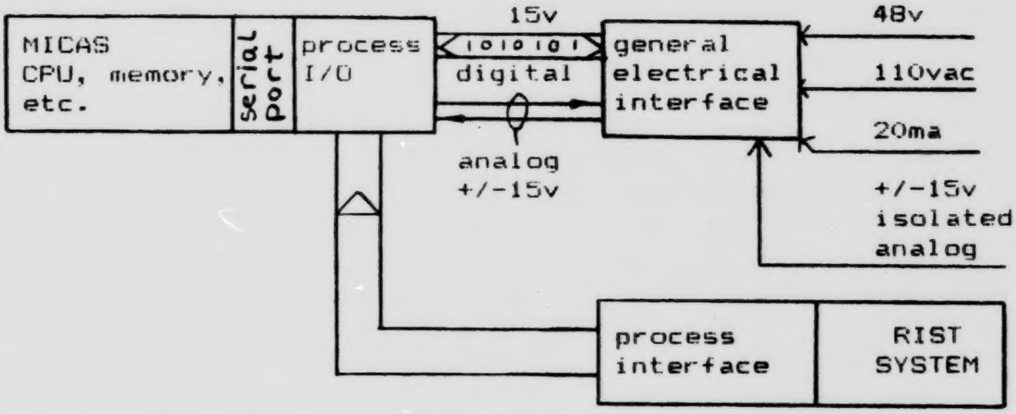


Fig. 2.2 MICAS/RIST General Interface Structure

The interface between the MICAS microprocessor control system and the RIST system established for the given project consists of nine digital input signals, four digital output signals, six analog input channels and two analog output channels. The interface with the specific input and output values is shown in fig. 2.3.

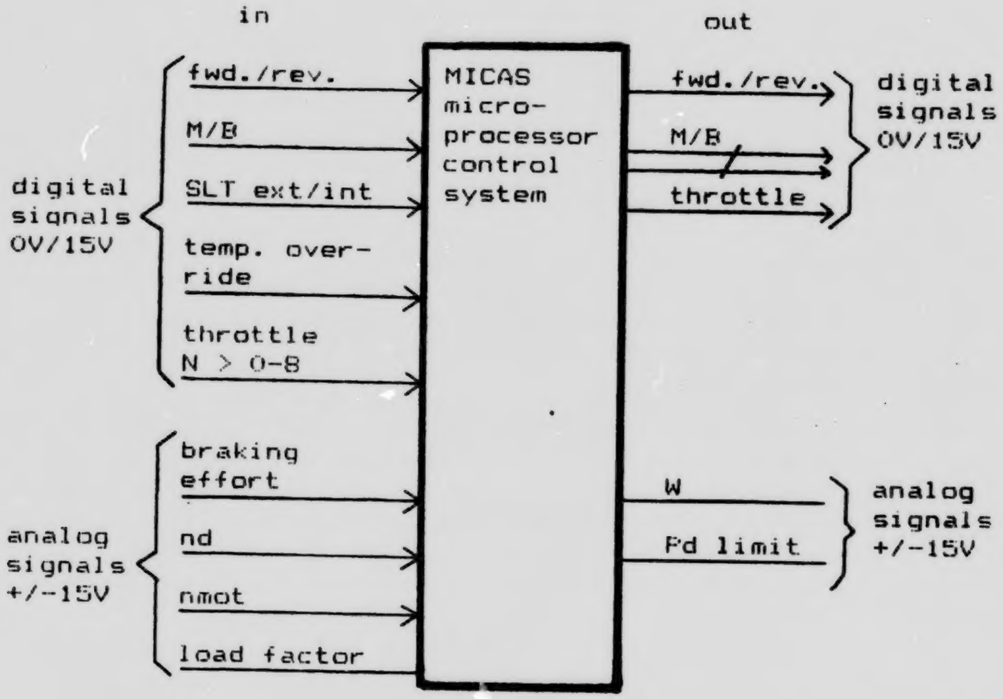

in
out
digital signals 0V/15V
fwd./rev.
M/B
SLT ext/int
temp. override
throttle N > 0-8
MICAS micro-processor control system
fwd./rev.
M/B
throttle
digital signals 0V/15V
analog signals +/-15V
braking effort
nd
nmot
load factor
W
Pd limit
analog signals +/-15V

Fig. 2.3 MICAS/RIST Interface with Input and Output Values

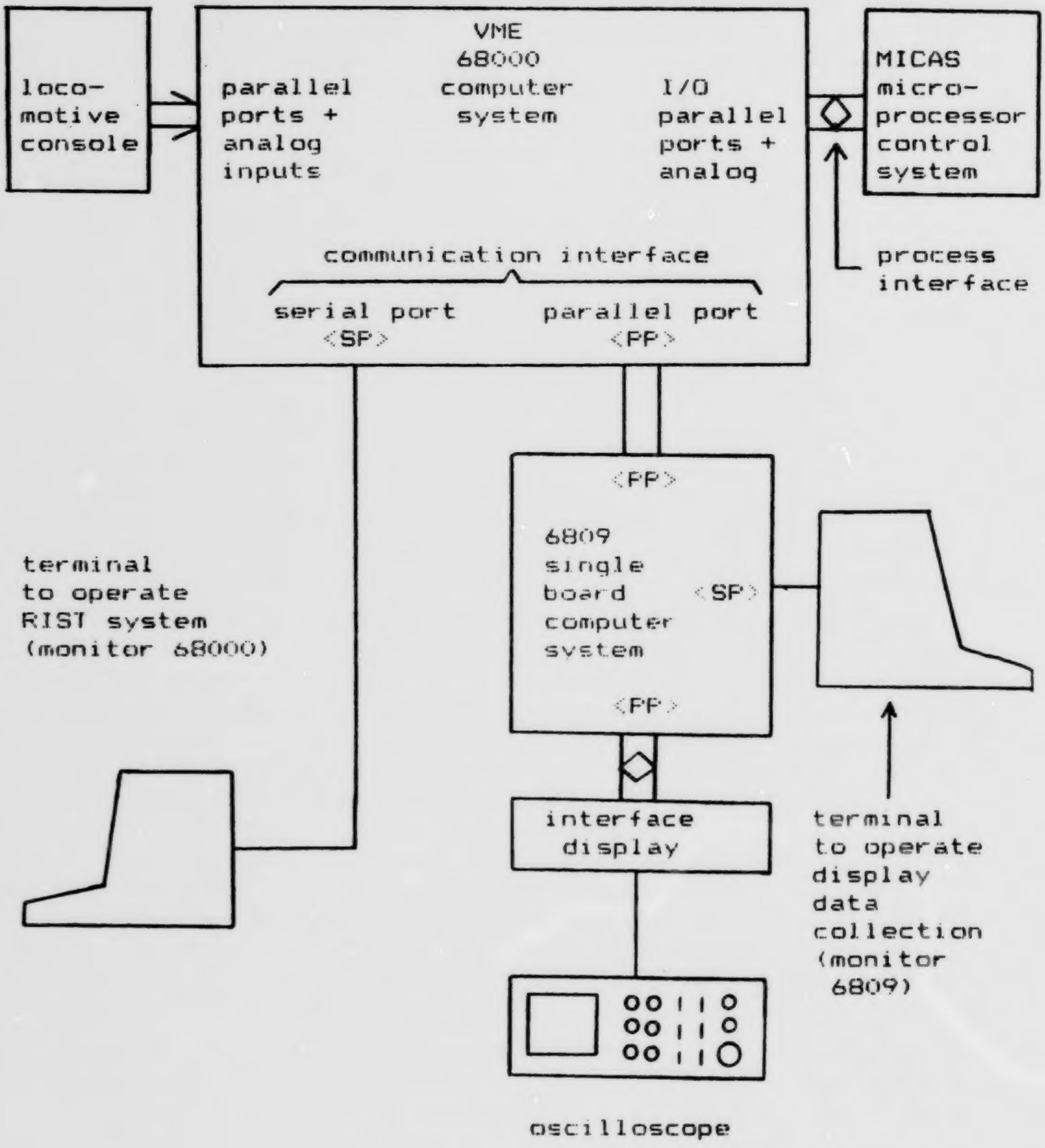
2.3.2 The hardware of the RIST system

Any computer which has a process interface can be used to build the RIST system. For the project two computers based on microprocessors were used. The simulation of the plant, the process interface to the MICAS microprocessor control system, the operator interface and part of the communication interface were implemented on a VME computer system from Motorola. The VME system is a based on the Motorola M68000 microprocessor operating at 8 MHz. It has an I/O system with two serial ports, two 32 Bit parallel ports, an analog input for 8 analog channels and an analog output for 4 channels.

The second computer used is the University of Toronto single board computer (SBC) based on the Motorola M6809 microprocessor. It is used as part of the communication interface. Its function in the RIST system is to collect data from the simulation and to display these data through a digital to analog interface on an oscilloscope. The oscilloscope is controlled by the display processor in such a way, that it looks like a six channel chart recorder. This allows one to view any variable in the simulated plant. In a later extension the display system was also connected to the MICAS system to display any internal variable of the control system.

The RIST system hardware is shown in fig.4. The simulation and testing is controlled from a terminal at the VME computer and with signals from the "locomotive console". The "locomotive console" is the interface to the plant operator and is a duplicate of the real console. It has all the command settings, available on the real locomotive and allows the interactive control of the system. The display is controlled from another terminal.

The use of two microcomputers is for two reasons. Using a single dedicated computer for the display makes the task easier to get a real time display on the oscilloscope. Beyond this the display interface to the oscillograph is very simple, so that all processing for the display is done in the computer, which requires its full time. The other reason is that the single board computer serves also as a support unit to program the MICAS computer. The two computer set up was the easiest way to implement the RIST system and part of the MICAS support system with the available resources.


loco-
motive
console
VME
68000
computer
system
parallel
ports +
analog
inputs
I/O
parallel
ports +
analog
MICAS
micro-
processor
control
system
process
interface
communication interface
serial port
<SP>
parallel port
<PP>
<PP>
6809
single
board
computer
system
<SP>
<PP>
terminal
to operate
RIST system
(monitor 68000)
interface
display
terminal
to operate
display
data
collection
(monitor
6809)
oscilloscope

Hardware RIST System

Fig. 2.4 RIST System Hardware

2.3.3 The software of the RIST system

The software consists mainly of several larger program modules. These are programs for the simulation and for the interfacing of the system to man and microprocessor controller.

The modules for the simulation of the locomotive system are the programs DLOCO,ELOCO,MLOCO,TLOCO and LOCO. The program DLOCO simulates the diesel engine. The program ELOCO simulates the traction command control. The program MLOCO simulates the electric drive with the inverter and the wheel-rail contact. The program TLOCO simulates the train. The program LOCO integrates all these modules and the program IOLOCO to the RIST system.
All the modules with the term 'LOCO' run on the VME computer. The programs are written in an extended version of the PASCAL programming language, which is available for the microprocessor development system used in the project.
The simulation programs DLOCO,ELOCO,MLOCO,TLOCO and LOCO are described in detail in chapter 3 together with the system analysis and the simulation.

The modules for the interfacing of the system are the programs IOLOCO,DATDIS2 and MDABL. These modules make the system interactive. They provide the software for the process interface to the MICAS microprocessor, for the plant operator or 'locomotive console interface and the communication interface. The communication interface has the two parts, the VME computer terminal and the data display interface as shown in fig.2.4.
The program IOLOCO contains the software for the process interface to the MICAS microprocessor as shown in fig. 2.3. and the software for the 'locomotive console' interface. It also contains the software for the interface to the VME computer terminal, which controls the simulation. A special routine IRQ3 in the program module IOLOCO provides the link to the data display system. The program module IOLOCO is the core of the software providing interactive simulation. The program is described in detail in section 2.3.4.

The program DATDIS2 runs on the SBC 6809. It is written in PASCAL. It contains the software for the display part of the communication interface.
The DATDIS2 display program has the task to display data on the oscilloscope, which represent variables of the system simulation and the control algorithms executed in the VME and MICAS computers. The program acquires these data from the VME and MICAS computers by interrupting those computers. The display program uses the acquired data to update its data set and then writes a new graph on the oscilloscope screen.
On interrupt the VME computer executes the routine IRQ3 in the program module IOLOCO and provides one Byte of data at the parallel port, which is connected to the display computer. The MICAS computer works similar, but it is connected through a serial port and a separate interrupt line to the display computer. The supporting software in the MICAS system is the program MADBL.
The program DATDIS2 is described in section 2.3.5.

The program MADBL is a 'MICAS block' designed and written in 8086 Assembler language. It provides an interrupt to the DATDIS2 program data for the display computer. The data are supplied through the serial port used for the monitor operation. The program is described in section 2.3.6.

2.3.4 The program module IOLOCO

A first demonstration of the use of the program is given in 3.2.3, where results of the diesel engine simulation are given. This module contains the software for all interfaces provided by the VME computer. It performs the following major tasks.
It sets up the parallel ports, the analog input ports and the analog output ports.
It provides communication at the process interface to the MICAS computer reading and writing signals to and from the ports connected to the MICAS. This includes the necessary data conversions between the internal data format for calculations and the data format required by the analog input and output ports.
It provides the interface to the locomotive console through the ports of the VME computer. This includes data format conversion and signal conditioning as well as debouncing for the command switches and the throttle settings on the console.
It serves interrupt requests from the display computer thus contributing to the display part of the communication interface.
It handles the complete communication with the VME computer terminal providing the interactive simulation control. >From the keyboard of the computer terminal it is possible to start the simulation, to select the time constant for the train simulation, to select the diesel load source and to switch the diesel overload protection on and off. The keyboard allows the selection of different traction characteristics and the control of the transition between selected characteristics to simulate changing traction conditions. Finally it is possible to select different data sets for the display from the keyboard. As indicated the display can show up to eight different variables, but the simulation has many more variables. To show these variables, they are grouped in display data sets of eight values. One of these data sets is then selected by the keyboard for the display. It is possible to change this selection at any time during the simulation.
A list of the keyboard commands to control the simulation is added to Appendix A1.

A short description of the procedures or subroutines of the program module IOLOCO and the the program listing is given in Appendix A2.

2.3.5 The program DATDIS2

A first demonstration of the use of the program is given in 3.2.3, where results of the diesel engine simulation are given. This program runs on the SBC 6809 computer. It is part of the communications interface.
The program displays a set of NV variables on the oscilloscope. The maximum number of variables NV is eight. This set of variables is selected by the keyboard. The numbers 1 to 8 select variables from the data set from the VME simulation computer. The numbers 1 to 4 with the prefix M select variables from the data set from the MICAS microprocessor control system.
The program collects MD sets of NV variables and puts them in the display buffer. The display buffer has a size of about 1000 = MD * NV. To update the display buffer the oldest set NV is deleted and a new set NV added.
The data of the display buffer are periodically written to the oscilloscope. The data points of the oldest set are at the left edge of the oscilloscope screen, the data points of the newest set are at the right edge. This has the effect, that the

screen of the oscilloscope looks like a chart recorder. The time proceeds from the left edge to the right. The newest data appear at the right edge and the oldest visible data disappear at the left edge. A display with three variables is shown in fig. 2.5. It shows the diesel speed, the diesel torque and the traction power command. The display just started, therefore the screen or 'chart' is not yet filled up to the left edge.

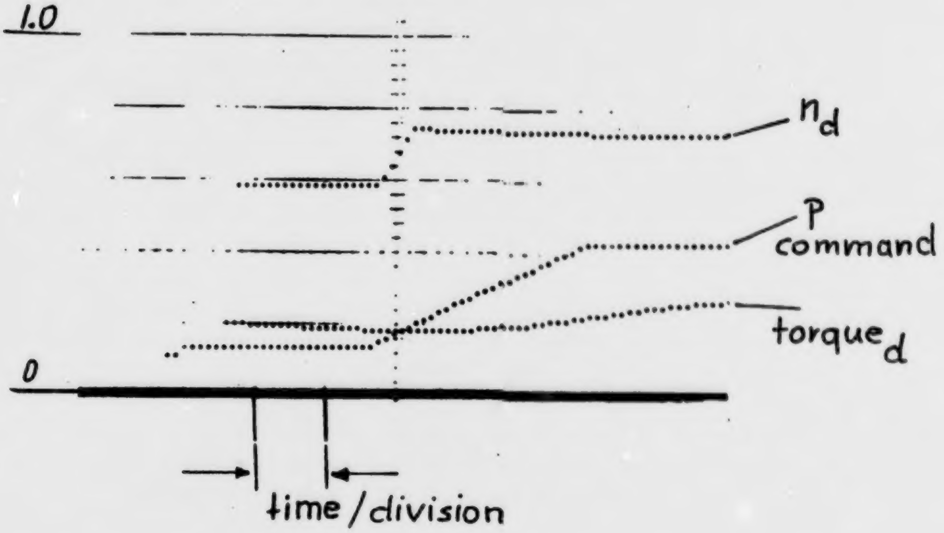


Fig. 2.5 Display with Three Variables

To study details the display can be stopped or 'frozen' by a keyboard command. To avoid a complicated synchronization system between the oscilloscope and the display computer a 1 Hz square wave clock is displayed on one of the channels. This clock can be used to calibrate the deflection in the time axis of the oscilloscope. A zoom effect can be achieved by freezing the display and changing the time scale at the oscilloscope.

The principle structure of the DATDIS2 program is shown in fig. 2.6. The loop is executed every 27 msec, which provides 37 displays per second. Thirty-seven displays per second results in a standing picture on the oscilloscope. The resulting sample rate of 37 Hz is sufficient for the studies done in this project. The resolution of the displayed data is 0.8% , resulting from the data word width of eight bits.
The program is written in PASCAL, which provides a good documentation of the details of the program. The program listing is attached in Appendix A3.

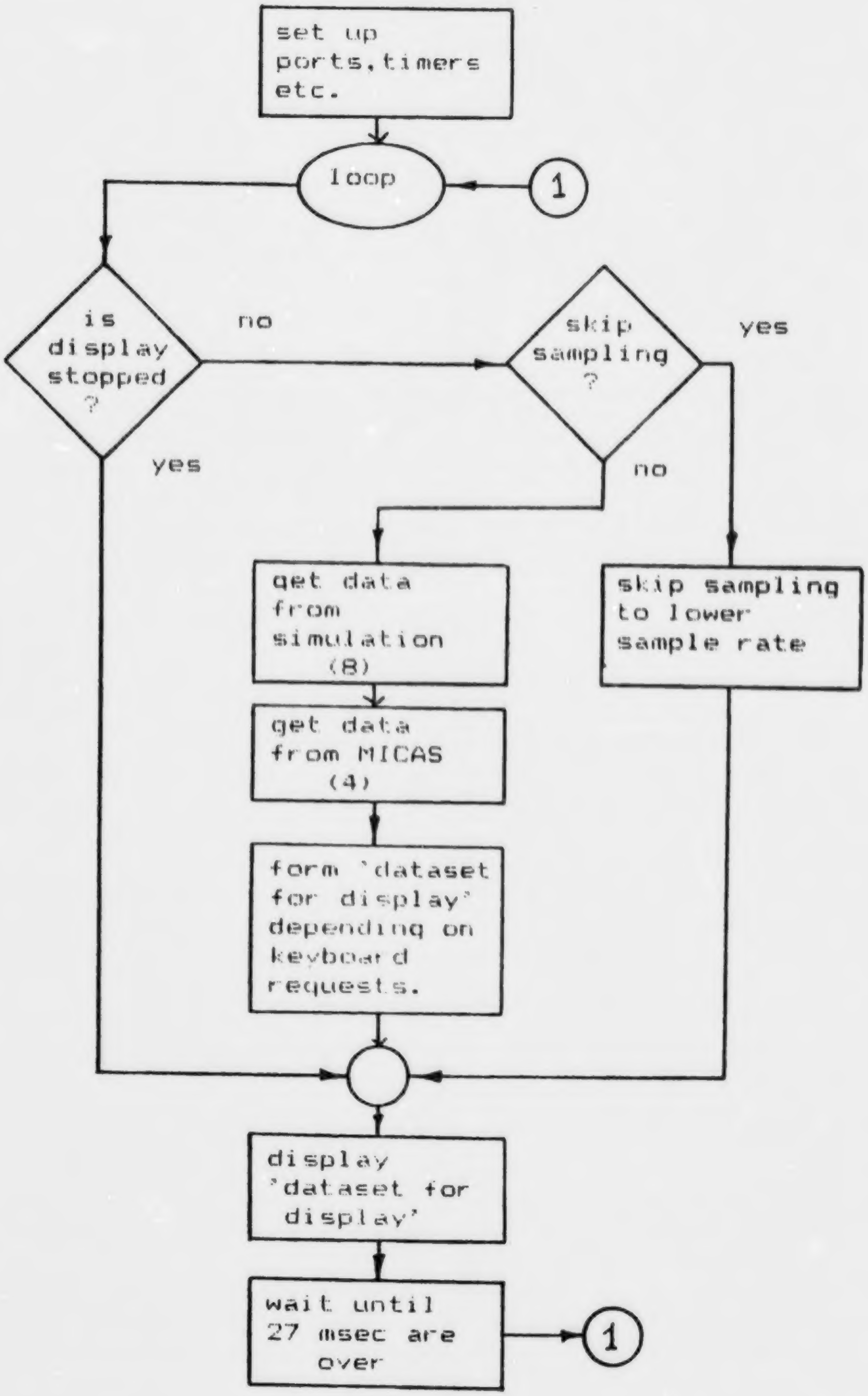

set up
ports,timers
etc.
loop
1
is
display
stopped
?
no
skip
sampling
?
yes
yes
no
get data
from
simulation
(8)
skip sampling
to lower
sample rate
get data
from MICAS
(4)
form 'dataset
for display'
depending on
keyboard
requests.
display
'dataset for
display'
wait until
27 msec are
over
1

Fig. 2.6 DATDIS2 Program Structure

2.3.6 The program modules MADB, MDABL

A first demonstration of the use of the program is given in 3.2.3, where results of the diesel engine simulation are given. These software modules are 'MICAS blocks' designed to run on the MICAS controller. The user summary and program listings can be found in Appendix A3. They provide the MICAS side interface to the SBC 6809 display computer. MDAB acquires and stores four selected channels of data in MICAS memory. MDABL is an "event triggered" routine that serves the interrupt requests from the display computer. Unlike the VME System - display computer, which employs a dedicated parallel data path, the MICAS interface uses the user serial communication link to send the display data to the SBC 6809. The MDAB modules, with a four channel configuration, increase the MICAS loading time by about 20%.

2.4 Support systems for programming

Support systems for programming the programs for the different computers VME, SBC6809 and MICAS are written in high level languages, assembler or in a special control block language for the MICAS system. This source code has to be converted to executable machine code and has to be put into the target computer.

The program development in this project was supported by a Hewlett Packard microprocessor development system HP64000. It provided all necessary compiler, assembler and linker programs for the different computers.

In the development stage the code is put into the target computer preferably by down loading the code into the computers RAM memory. This allows easy code changes during the test and debugging phase.

The down loading is done through serial communication links between the development system and the processors. For this the target processor needs a communication port and a basic software program (monitor) to be able to communicate with the down loading system.

In fig. 2.7 the principle connection between the computer systems is shown. The support for the different computers is described in the paragraphs below.

2.4.1 Support for the VME computer

The computer has a basic monitor program, which allows downloading of machine code in Motorola S-record format to the VME system. The monitor starts the loaded program and provides basic tests like single stepping and run to a break point. The VME M68000 was programmed in extended PASCAL available from Hewlett Packard on the HP64000 development system. Only three small routines for arithmetic were written in 68000 assembler. A host Pascal program available on the development system was used to generate look-up-tables used in the simulation program.

The assembler is available on the development system. The source programs were compiled, assembled and linked on the HP64000 and then downloaded to the VME system by the communication program of the HP64000 system.

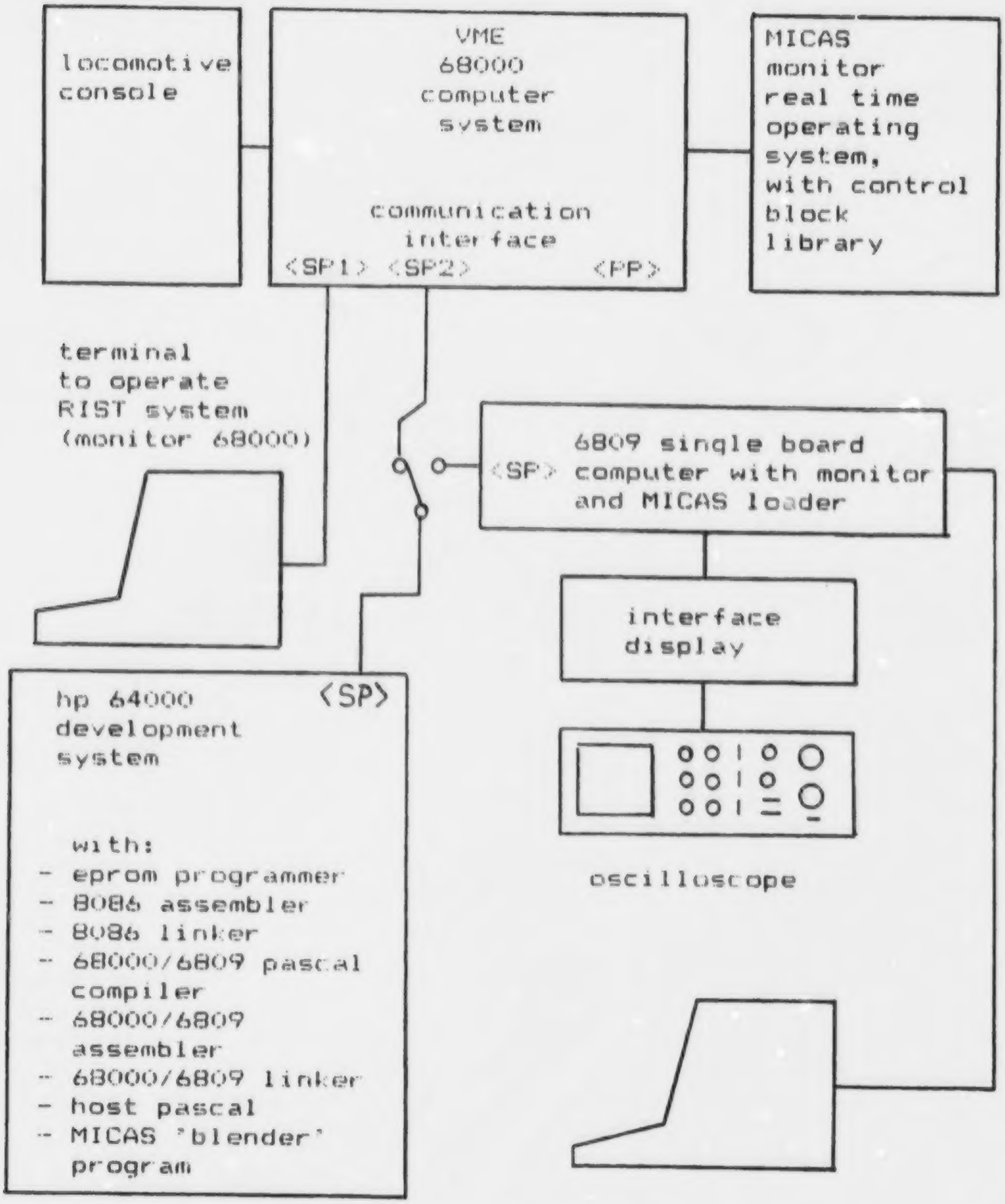

locomotive console
VME 68000 computer system
communication interface
<SP1> <SP2> <PP>
MICAS monitor real time operating system, with control block library
terminal to operate RIST system (monitor 68000)
<SP> 6809 single board computer with monitor and MICAS loader
interface display
oscilloscope
hp 64000 development system <SP>
with:
- eprom programmer
- 8086 assembler
- 8086 linker
- 68000/6809 pascal compiler
- 68000/6809 assembler
- 68000/6809 linker
- host pascal
- MICAS 'blender' program

Fig. 2.7 Connection Between Computer Systems

2.4.2 Support for the single board computer SBC 6809

The computer has a monitor and a versatile communication program, which allows one to receive and send code in different formats. For the display function the SBC 6809 was programmed in extended PASCAL and 6809 assembler. For the MICAS support a program was written in extended PASCAL. The compilers, assembler and linker were available on the HP64000 development system.

2.4.3 Support for the MICAS computer

The system as made available for the project has three software packages on board in EPROM memory: A basic monitor, which allows one to communicate with the system through a serial port. A real time operating system which contains also a library of basic control programs. An application program for a locomotive control. The basic control programs or blocks are connected by the application program to form the final executable control program.
Besides this software and the MICAS computer itself, there was no support or programming system to support the work with the MICAS microprocessor control system. Therefore support systems had to be made available for the following tasks.

A terminal is needed to operate the MICAS monitor to check and alter RAM memory. The SBC 6809 with its communication program and terminal is used as the human interface to use the monitor of the MICAS system.
To write and test an application program a facility is required to edit the program and then to load it into the RAM memory of the MICAS system. The SBC 6809 computer has a software package TEACH, which can edit and handle files. TEACH was used to edit MICAS application programs. To download the developed programs a MICAS loader program MLDR was written for the SBC 6809 computer. This program takes the MICAS program from the SBC 6809 memory and puts it into the MICAS RAM memory. The MLDR program uses for this the communication facilities of the SBC computer and of the MICAS computer and its monitor. (see Appendix A4)
An extension of the MICAS loader program MLDR allows also to edit MICAS application programs on the HP64000 development system send them to the SBC computer and load them then into the MICAS computer.
A facility was needed to transfer the provided application programs from EPROM memory into RAM memory. This is necessary for the testing, as program alterations can be made easily only in RAM memory. The EPROM handling facilities of the HP64000 system were used to read the EPROMs. A program MICAS code blender was written in host Pascal to convert the code from the EPROMs in code suitable for downloading to the MICAS RAM memory. The downloading from the development system to the MICAS computer is then done through the SBC computer with the support of the MICAS loader program MLDR.
To alter the MICAS operating system and to add new control blocks, it is necessary to change the EPROMs in the MICAS system. This was done with the EPROM handling facility of the development system.

The programming of new MICAS control blocks had to be done in Intel 8086 assembler, as the microprocessor used in the MICAS mcs is the Intel 8086. This programming was done on the HP development system using the 8086 assembler and linker. The code was then transferred to EPROMs, that were put into the MICAS system.

3. Locomotive System Simulation

3.1 Introduction

The locomotive system is simulated to verify the system analysis and the control algorithms developed from this analysis. The simulation is done with the Real time Interactive Simulation and Test (RIST) system. The RIST system is implemented on the VME Motorola M68000 computer. The programs for the simulation are designed and written in PASCAL.
The simulation program consists of the modules DLOCO,ELOCO,MLOCO, ASMCURVE and TLOCO. These modules and the module IOLOCO for the interfacing are execute by the program LOCO.

The following sections will provide a description of the simulation of the diesel engine, the electric traction system, the wheel rail contact and the train characteristic. Fig. 3.1 gives an overview of the simulation and its different parts and programs.

The system is specified first, then the model is derived and the block structure is described. Finally the program or program module is presented and examples demonstrate the function of the simulation.
The program LOCO will be described separately in section 3.5. This section will also discuss some aspects of real time simulation and the algorithms used.

3.2 Simulation of the diesel engine operation

3.2.1 System description and modelling of the diesel engine and its controls

The diesel engine is modelled as a first order lag system with an inertia and a friction. The diesel is loaded by a load torque at the shaft. The friction has a constant and a speed proportional term. The diesel engine operates only above a certain minimum speed and stalls if the speed drops below this minimum. The diesel engine is started by a start command, which can be activated only if the locomotive controller is in notch one. This basic structure is shown in the block diagram in fig. 3.2.

The diesel engine is controlled by the throttle, which has normally eight notches. A notch position provides two pieces of information for the operation of the diesel engine. The first is a speed command for the diesel engine speed. The second is a value for the maximum traction power available in this notch. This value together with the maximum auxiliary power required from the diesel in a certain notch provides the information what maximum diesel shaft power will be available from the diesel for a given notch setting. The maximum diesel shaft torque available for a certain notch can be calculated from the diesel speed and the maximum diesel shaft power.

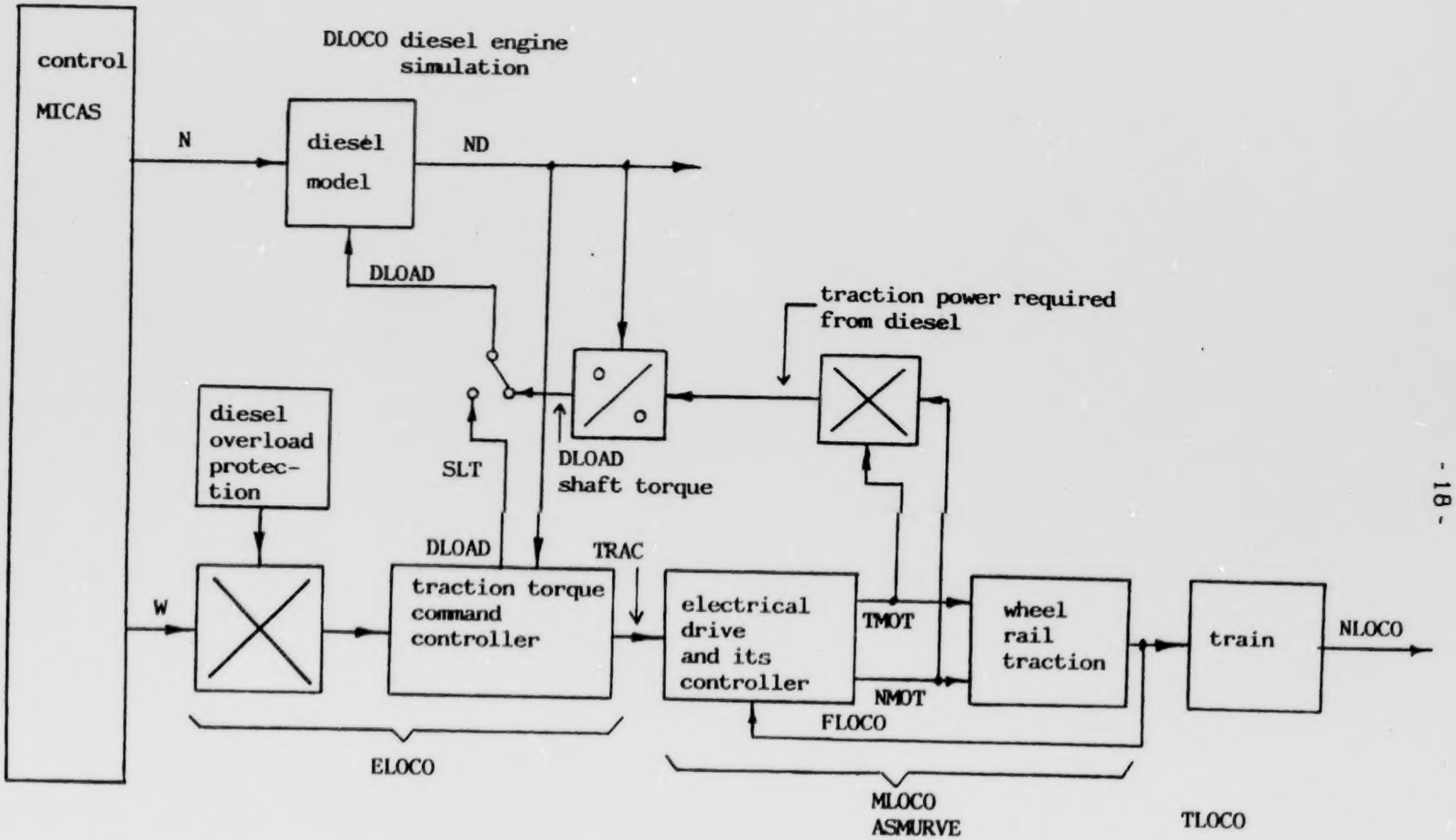

control
MICAS
N
W
DLOCO diesel engine simulation
diesel model
ND
DLOAD
diesel overload protection
SLT
DLOAD
shaft torque
DLOAD
TRAC
traction torque command controller
traction power required from diesel
electrical drive and its controller
TMOT
NMOT
FLOCO
wheel rail traction
train
NLOCO
ELOCO
MLOCO
ASMURVE
TLOCO

Fig. 3.1   Diesel Engine Operation Block Diagram

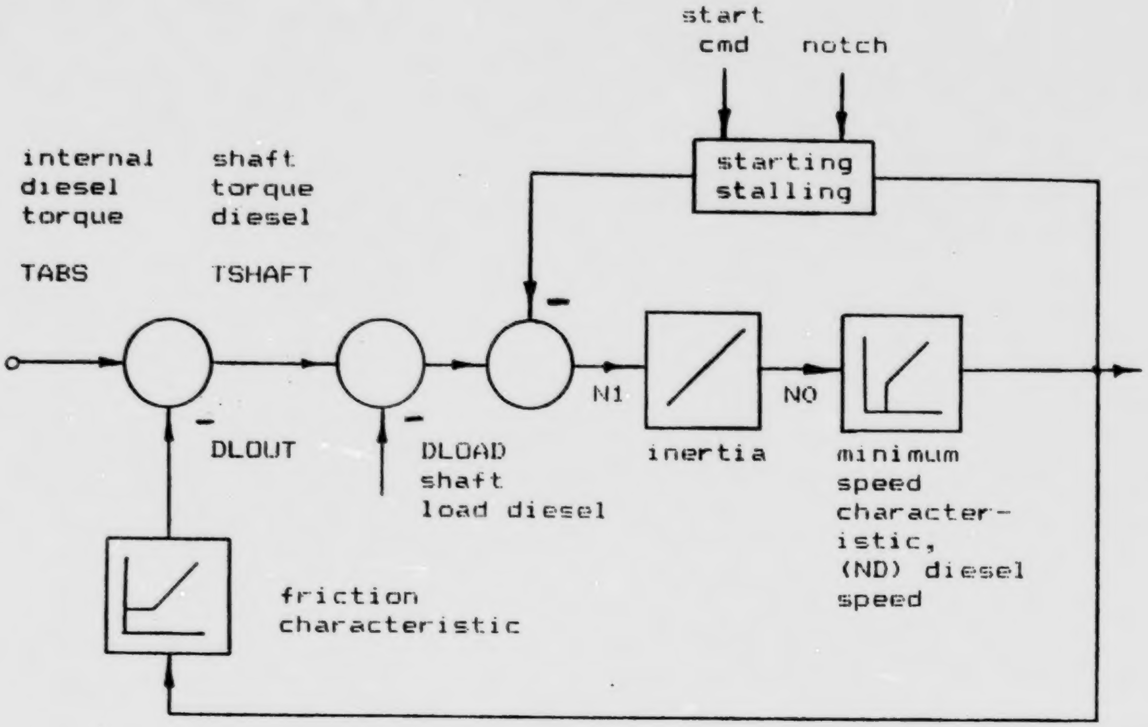

start cmd
notch
starting stalling
internal diesel torque
TABS
shaft torque diesel
TSHAFT
N1
NO
inertia
minimum speed characteristic, (ND) diesel speed
DLOUT
DLOAD shaft load diesel
friction characteristic

Fig. 3.2 Diesel Engine Basic Control Structure

The diesel control is modelled assigning to the notch setting of the throttle a diesel speed and a maximum diesel shaft torque. For the speed control a closed loop control with a proportional speed regulator is used. In this loop the diesel is modelled as a proportional function with a limit on the shaft torque. The limit is variable and depends on the notch setting. The speed command to the diesel engine is normally limited in its rate of change. This is modelled by a rate limiter in the diesel speed command channel. The structure of the diesel control is shown in fig. 3.3 using the basic structure from fig. 3.2.

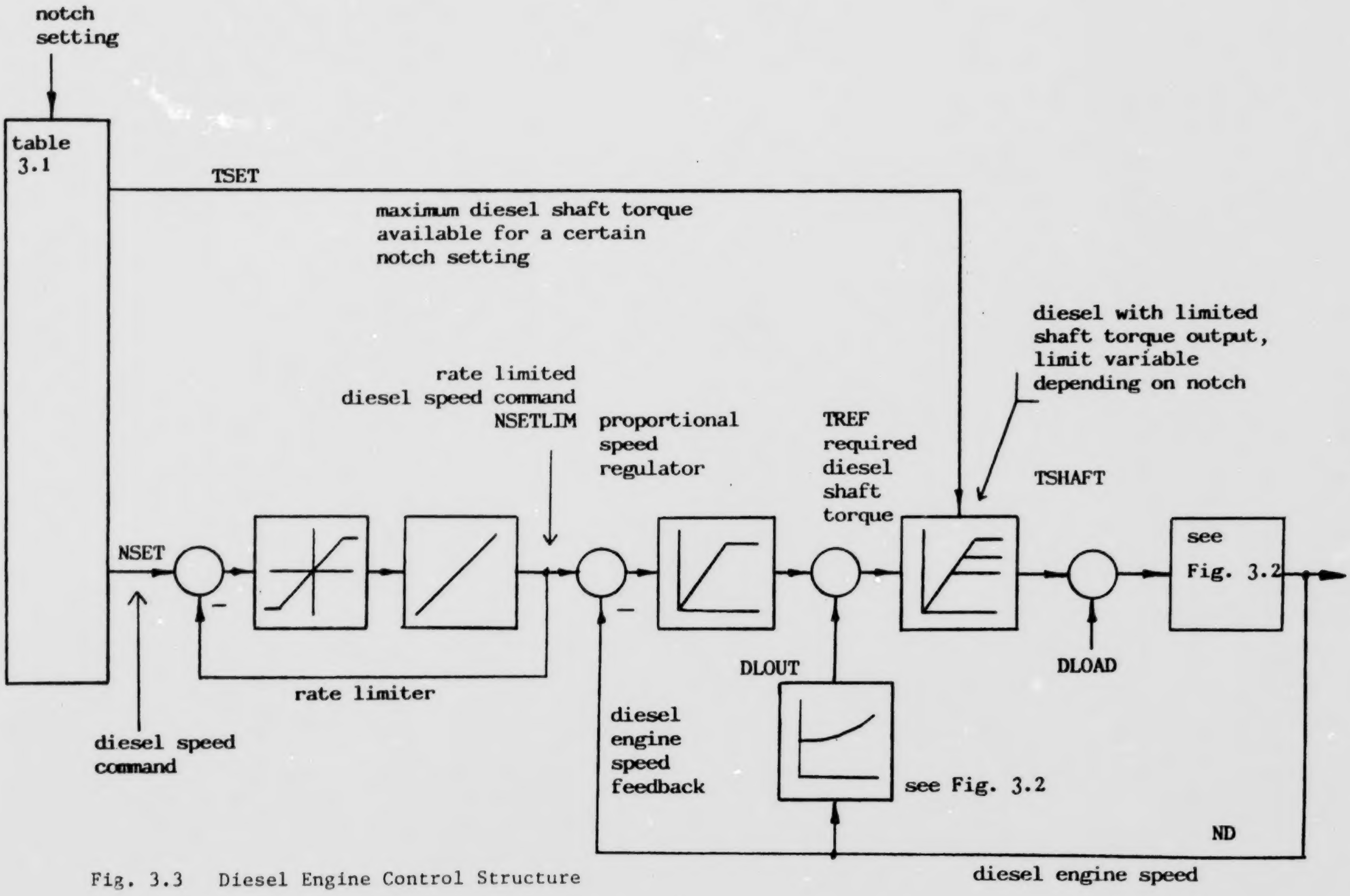

notch setting
table 3.1
TSET
maximum diesel shaft torque available for a certain notch setting
NSET
diesel speed command
rate limiter
rate limited diesel speed command NSETLIM
proportional speed regulator
diesel engine speed feedback
TREF required diesel shaft torque
DLOUT
see Fig. 3.2
diesel with limited shaft torque output, limit variable depending on notch
TSHAFT
DLOAD
see Fig. 3.2
ND
diesel engine speed

Fig. 3.3 Diesel Engine Control Structure

The model in fig. 3.2. presents a diesel engine without a charger. The diesel provides shaft torque immediately, when it is requested at the shaft by the load (DLOAD). But many diesel engines are equipped with a turbo charger. A diesel engine with a turbo charger is characterized by the fact, that the shaft torque builds up slowly within 10 to 20 seconds. This dynamic property must be included in the diesel model, if transients from the wheel rail contact, like slipping, are to be studied. The charger is contributing to the torque only in the higher notches.

The turbo charger process is in principle a positive feedback loop. Depending on the actual shaft torque and the diesel speed exhausted gas is produced, that drives the charger, which in turn increases the shaft torque. The process is unstable and operates finally at a constant operating point only, as a limiter in the feedback loop interrupts the process. The principle structure to describe the turbo charger is shown in fig. 3.4.

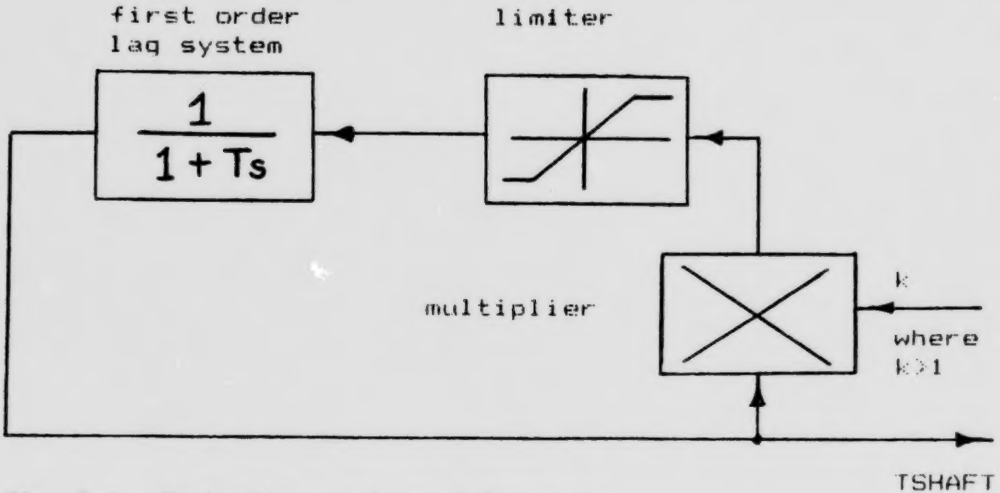


Fig. 3.4 Turbo-Charger Control Structure

The differential equation describing the process is:

$$TSHAFT + d(TSHAFT)/dt \times T = k \times TSHAFT$$

with the solution

$$TSHAFT = \exp((k-1) \times t/T) + constant$$

The output increases for $k > 1$. The process reaches an equilibrium if the differential gain in the loop is reduced to one by the limiter.
This principle structure is used in the diesel model, as shown in fig. 3.5.

The turbo charger loop is acting on the variable limiter representing the diesel and its output shaft torque. In steady state the output from the charger loop TDELAY is equal to the maximum available diesel shaft torque TSET.

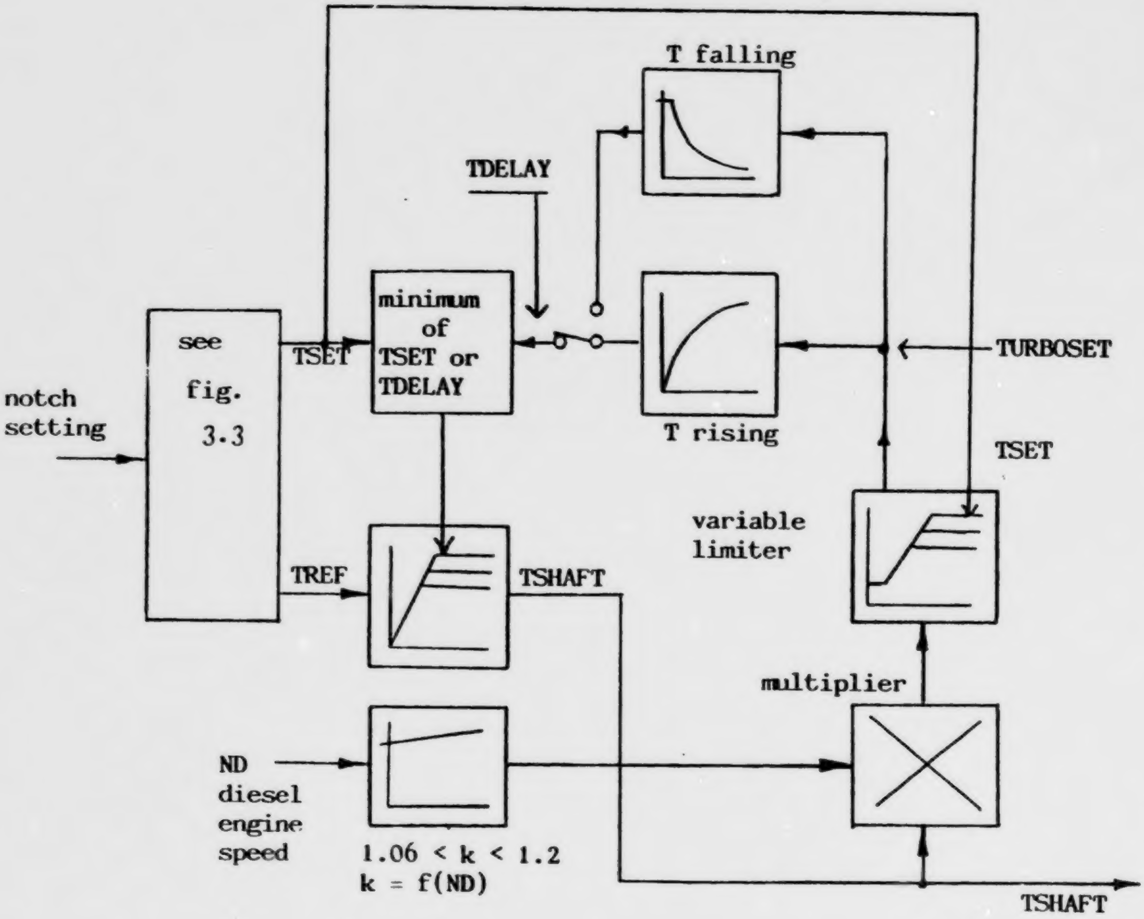

T falling
TDELAY
minimum of TSET or TDELAY
see fig. 3.3
TSET
notch setting
T rising
TURBOSET
TSET
variable limiter
TREF
TSHAFT
multiplier
ND diesel engine speed
1.06 < k < 1.2
k = f(ND)
TSHAFT

Fig. 3.5 Diesel Engine Model Diagram

During transients TDELAY rises only slowly and is limiting TSHAFT to a lower value than TSET. To reflect the property of the charger, that the torque rises slower than it falls, two different time constants are used in the charger loop. The charger loop gain depends on the diesel speed, reflecting the fact, that the diesel responds faster at higher speed.

3.2.2 Quantities of the diesel engine model

To study the diesel engine behavior quantitative values had to be assigned to the model. A 4000 hp locomotive was chosen to supply realistic quantities. The locomotive has a throttle with the notch positions idle and one to eight for traction. A certain diesel engine speed and traction power is assigned to each notch position. An assumption was made about the maximum auxiliary power required in each notch. It was assumed, that the maximum auxiliary power rises from about 210 KW in notch one to 400 KW in notch five and stays then constant up to notch eight. Table 3.1 shows the values used for the simulation in the project.

Diesel Engine Data for Simulation

throttle notch	traction power kW	diesel engine speed rpm	diesel engine power kW	diesel engine torque Nm	NSET diesel engine speed per unit	TSET diesel engine torque per unit
idle	-	400	-	-	0.364	-
1	36.6	400	251	5,992	0.364	0.188
2	103	490	376	7,328	0.445	0.230
3	236	580	526	8,660	0.527	0.272
4	413	680	712	9,999	0.618	0.314
5	765	770	1165	14,448	0.700	0.454
6	1323	860	1723	19,132	0.782	0.601
7	1984	960	2384	23,714	0.873	0.745
8	2984	1050	3384	30,776	0.955	0.967

Table 3.1

The maximum available diesel shaft torque is derived from the diesel speed and the maximum required shaft diesel power.

For the simulation all values used were per unit values. The per unit value for the diesel speed is chosen to 1100 rpm and the per unit value for the diesel power is chosen to 11,000KW/3. The maximum per unit diesel shaft torque calculates from this to 31,831 Nm. The resulting per unit values for the diesel speed and maximum available diesel shaft torque are shown in the last columns of table 3.1 and in fig 3.6. These values are the input control values for the diesel model NSET and TSET shown in fig. 3.3.

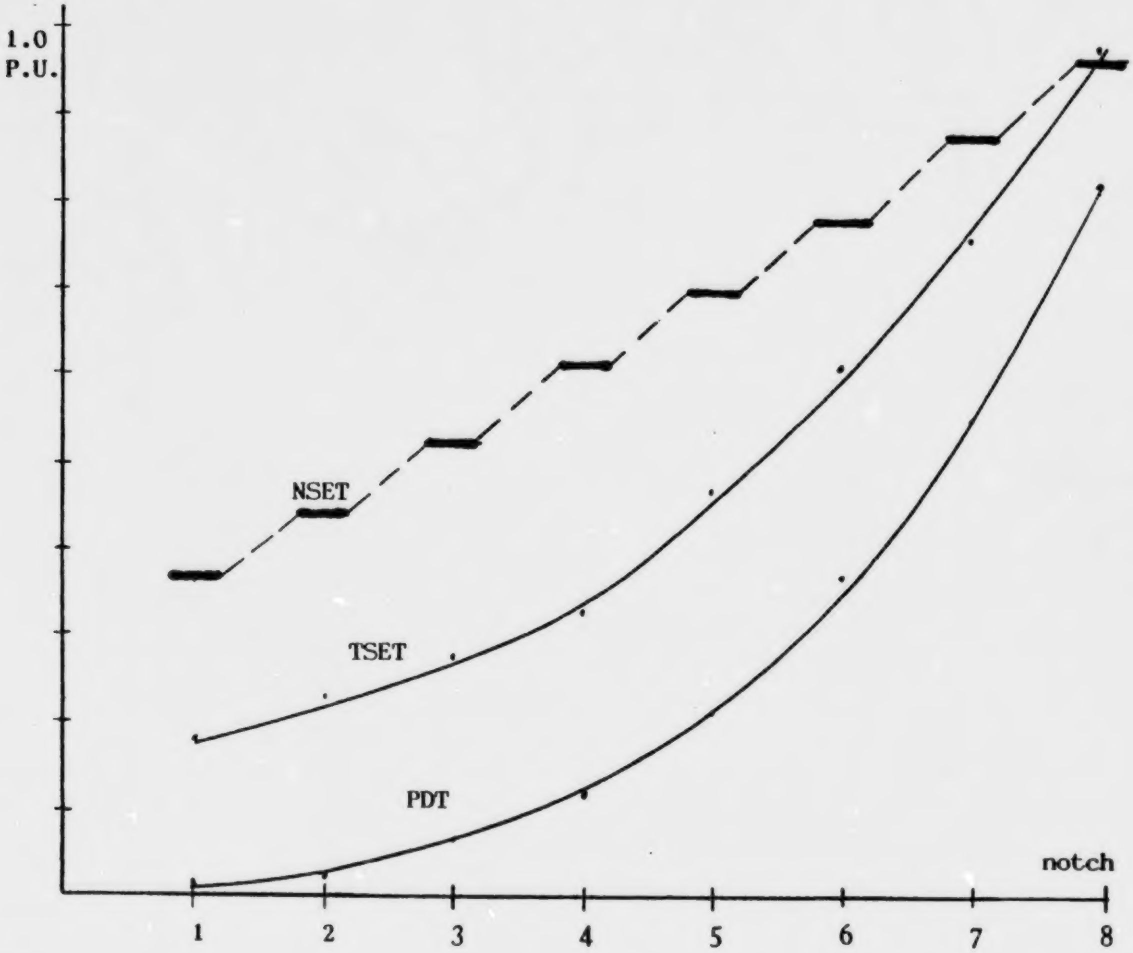


Fig. 3.6 Engine Simulation Data

The complete model with its values is shown in fig. 3.7.

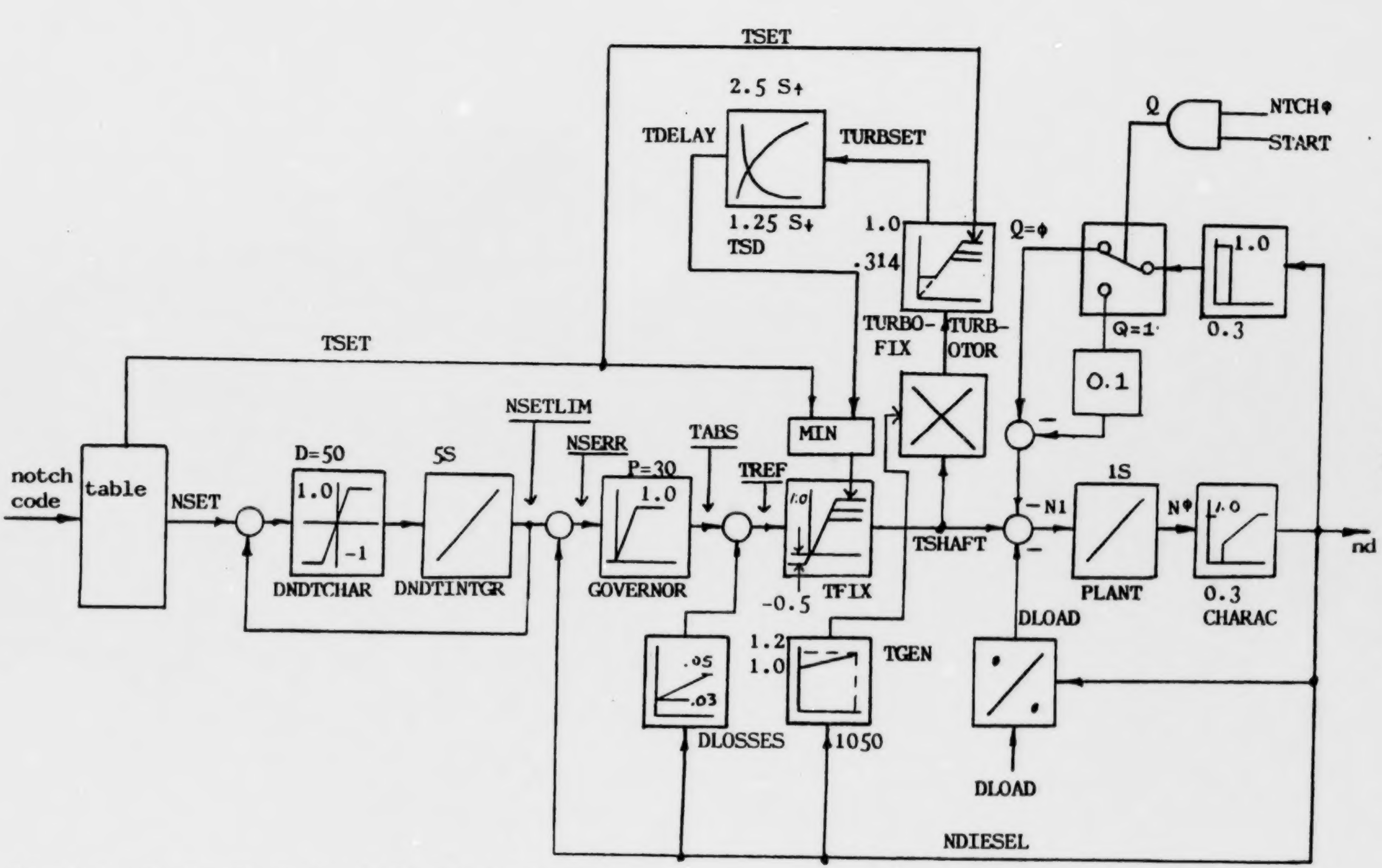

TSET
2.5 S+
TDELAY
TURBSET
1.25 S+
TSD
1.0
.314
Q
NTCHΦ
START
Q=Φ
1.0
Q=1
0.3
TURBO-
FIX
TURB-
OTOR
0.1
TSET
NSETLIM
NSERR
TABS
TREF
MIN
notch
code
table
NSET
D=50
1.0
-1
5S
P=30
1.0
1:0
-N1
1S
NΦ
1.0
nd
DNDTCHAR
DNDTINTGR
GOVERNOR
-0.5
TFIX
TSHAFT
PLANT
0.3
CHARAC
DLOAD
.05
.03
1.2
1.0
TGEN
DLOSSES
1050
DLOAD
NDIESEL

Fig. 3.7 Engine Simulation Model Block Diagram

Beyond the table 3.1 describing the relation of the notches to the diesel speed and the maximum diesel shaft torque the following assumptions are made:
The diesel loss torque is $DLOUT = 0.03 + 0.02 \times ND$.
The diesel stalls, if the speed falls below 80% of the idle speed.
Starting the diesel is only possible, if the throttle is in notch one.
The diesel inertia is such, that the per unit shaft torque accelerates the diesel without shaft load from zero speed to per unit diesel speed in one second.
The diesel speed regulator is a proportional regulator with a constant gain of thirty. Its output is limited to positive values, assuming there is no negative or braking internal torque from the diesel.
The rate limiter for the diesel speed command is set to a rate of one per unit speed within five seconds.
The turbo charger is assumed to have no influence for throttle position of notch four or smaller. The delay time for rising torque is chosen to 2.5 seconds and the time for falling torque is half of this.
The speed influence is modelled as $k = 1.06 + 0.14 \times ND$.
The resulting time constant $T/(k-1)$ in the exponential function is then between 13 and 16 seconds.

3.2.3 Program DLOCO and examples of the diesel engine simulation

The program module DLOCO is executed by the program LOCO. The module DLOCO consists of several procedures or subroutines. A short description of these procedures and the listing of the program module is provided in the Appendix A2.
The block diagram of the simulation was shown in fig.3.7. The following fig. 3.8 and 3.9 demonstrate some aspects of the diesel engine simulation and verify the function of the real time simulation of the diesel engine part of the RIST system.

The figures give an impression of how the communication interface of the RIST system works. The figures were taken from the oscilloscope used in the data display system working with the program module IOLOCO and DATDIS2. The desired variables to be observed are selected, the time scale for the display set and the transient is started. To study the transient the display is stopped and if desired a photo for documentation can be taken.

Description of the simulation fig. 3.8. The simulated events demonstrate the function of the speed command rate of change limiter, the speed control loop, the torque response of the diesel with and without turbo charger influence and the stalling of the diesel engine at overload.
The figure shows the diesel simulation variables NSETLIM,ND,TSHAFT and TDLOAD. NSETLIM is the diesel engine speed command after the rate limiter. ND is the diesel engine speed. TSHAFT is the diesel engine shaft torque and TDLOAD is the torque loading the diesel shaft. The variables are shown in per unit values. The vertical axis gives about 0.2 per unit per division. The horizontal axis is the time, it is selected to 2 sec per division for this display.
The simulation shows a sequence of events. At the time $t = 0$ the diesel is operating in throttle position $N = 0$, that is 0.364 per unit diesel engine speed. There is no load applied to the diesel shaft, therefore TDLOAD and TSHAFT are zero.

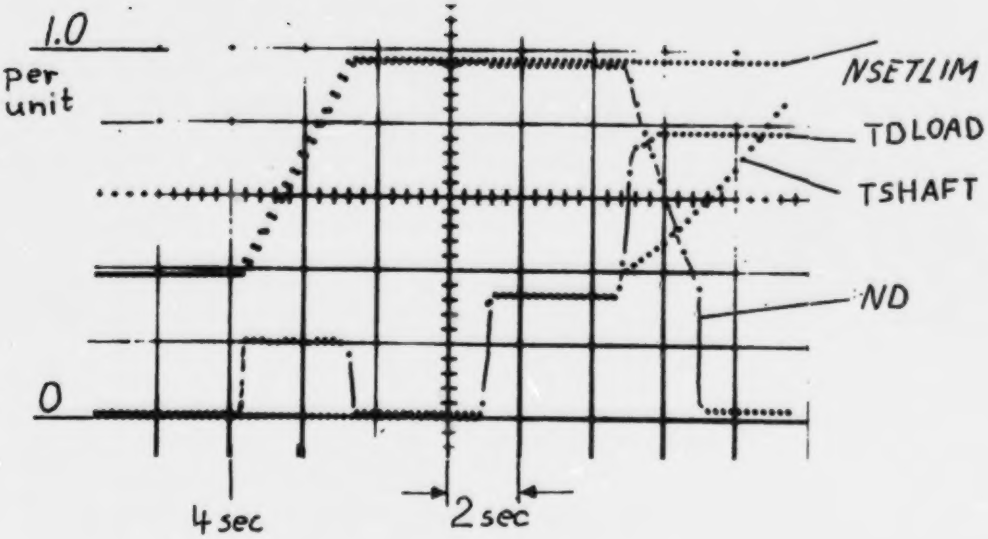


Fig. 3.8 Example of Diesel Simulation Variables

At t = 4 sec the throttle is suddenly advanced to notch N = 8. The diesel engine speed command NSETLIM rises with the maximum rate of the limiter of 0.2 units per second until it reaches the setting of notch eight, which is 0.955 per unit at about t = 7 sec. The value NSETLIM stays then constant.

The diesel engine speed tracks the NSETLIM command with a small delay. It reaches the required speed shortly after the command. During the acceleration of the diesel engine the governor in the speed control loop request a shaft torque of 0.2 per unit, which is sufficient to accelerate the unloaded diesel engine at this rate.

At t = 11 sec a load torque TDLOAD of 0.32 per unit is suddenly applied to the diesel engine shaft. The diesel engine shaft torque TSHAFT jumps immediately to the same value to balance the load. The diesel engine speed drops slightly, as the speed regulator is a proportional type of regulator, which has a small steady state error under loading. The immediate increase of the diesel engine torque is possible, as the required torque is in a range, where the turbo charger has not yet an influence.

At about t = 15 sec the load torque TDLOAD is suddenly raised to 0.79 per unit. The diesel engine shaft torque can not follow, as the turbo charged diesel engine reacts only slowly to this torque request. The slow rise can be clearly seen. The diesel engine is now grossly overloaded and the diesel engine speed drops within 2 sec to the idle diesel engine speed of notch zero and falls then to zero, as the diesel engine stalls. That the variable TSHAFT keeps on rising after the diesel stalled is a meaningless property of the simulation, as all diesel values after stalling have no meaning any more.

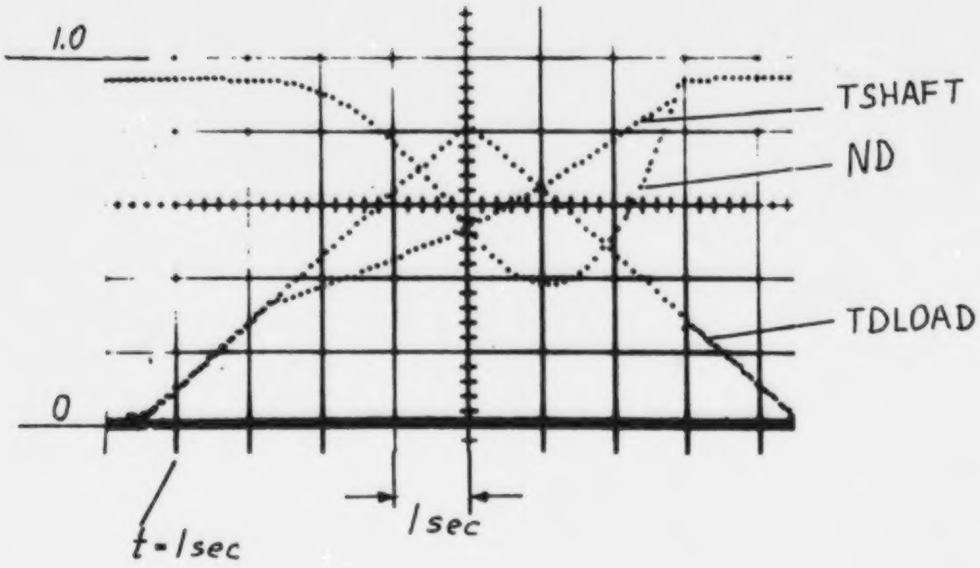


Fig. 3.9 Example of Diesel Simulation Variables

Description of the simulation fig. 3.9. The simulated events demonstrate the function of the influence of the turbo charger on the torque build up at the diesel engine shaft.

The figure shows the diesel engine simulation variables ND, TDLOAD and TSHAFT. ND is the diesel engine speed. TSHAFT is the diesel engine shaft torque and TDLOAD is the torque loading the diesel shaft. The variables are shown in per unit values. The vertical axis gives 0.2 per unit per division. The horizontal axis is the time, it is selected to 1 sec per division for this display.

The simulation shows a sequence of events. At the time t = 0 the diesel is operating with the throttle at the notch position eight. The per unit diesel engine speed is according to this 0.955 per unit. There is no load torque applied to the diesel engine shaft.

At t = 1 sec a load torque TDLOAD is applied. The load torque increases with 0.18 per unit per second until it reaches 0.8 per unit at t = 5 sec, then it drops with the same rate to zero again.

The diesel engine shaft torque follows the load torque in the beginning until about 0.34 per unit at t = 2.4 sec. In this range the turbo charger has no influence and the diesel engine can provide torque immediately. The diesel engine speed drops slightly at t = 1.8 sec due to the loading.

At t = 2.4 sec the torque reaches the range, where the turbo charging process gets influence on the torque build up. The load torque still rises with a rate of 0.18 per unit and second. The diesel engine torque in the turbo charger range can only rise with 0.07 to 0.09 per unit and second. As a result the diesel engine is overloaded and the diesel engine speed drops. This ends at t = 6 sec, when the disel engine torque exceeds the decreasing load torque and the diesel engine recovers from the overload.

The shape of the diesel engine shaft torque TSHAFT with the increasing rate of change demonstrates the positive feedback structure of the turbo charger. At about t = 8 sec the diesel reaches it set speed and the governor immediately lowers the shaft torque to 0.3 per unit just balancing the load torque.

The examples clearly demonstrate the type of diesel model established in the simulation system. In particular the modelling of the turbo charger influence is of importance as it limits the response of the diesel to fast transients. The model will allow one to recognize such transients, if they should occur i.e. as a result of a wrongly designed control of the electric drive.

3.3 Simulation of the electric drive

3.3.1 System description and modelling of the electric drive and its controls

The electric drive system consists of three major sections. This are the electric motors, the motor frequency controller or inverter and the traction torque command control. The traction torque command control derives the traction torque command from the traction power command such, that neither diesel nor electric motors are overloaded.

The motors of the locomotive are squirrel cage induction motors. They are fed from a single bus. The bus determines the stator frequency and the stator voltage of the induction motors. It is known from experience, that this locomotive cannot have single axle slipping. This allows one to model all motors together as one big equivalent traction motor. Clearly this model does not permit to simulate load sharing between the motors, but these studies have been done and were not the object of this project.
The equivalent traction motor is modelled as a single order lag system with inertia and motor losses. The electrical transients are modelled together with the inverter transients in a first order lag system at the input of the inverter. The electrical properties of the motor are then represented by a characteristic. The characteristic provides the internal motor torque as function of the rotor frequency. The parameter at these characteristics is the strength of the field of the motor. The field depends on the relation between stator voltage and stator frequency. This relation is programmed into the inverter control and therefore it is possible to use the stator frequency as parameter in the motor characteristic instead of the stator voltage or the field. In this way the stator frequency becomes the control input variable of the traction motor. The discussed simulation is shown in fig. 3.10.

The inverter generates the stator frequency from the motor speed and the rotor frequency. The rotor frequency is derived from the inverse torque rotor frequency characteristic used in the traction motor. The motor speed is the average speed of all locomotive axles. The input of the inverter characteristic is provided from a first order lag system. This block models all electrical transients in the torque producing system motor plus inverter. The block diagram of the inverter simulation is shown in fig. 3.11.

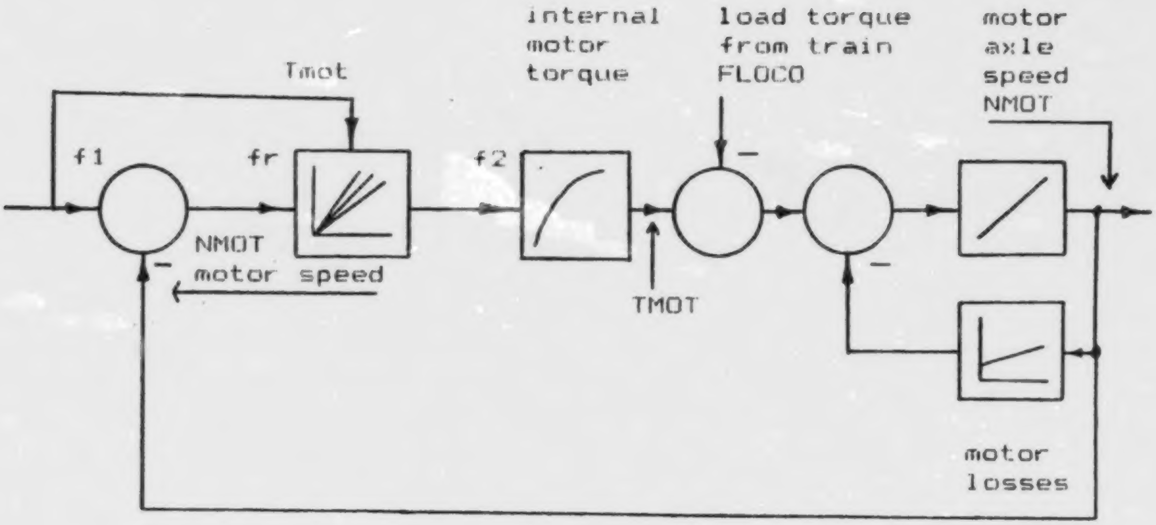

internal
motor
torque
load torque
from train
FLOCO
motor
axle
speed
NMOT
Tmot
f1
fr
f2
NMOT
motor speed
TMOT
motor
losses

Fig. 3.10 Drive Simulation Diagram

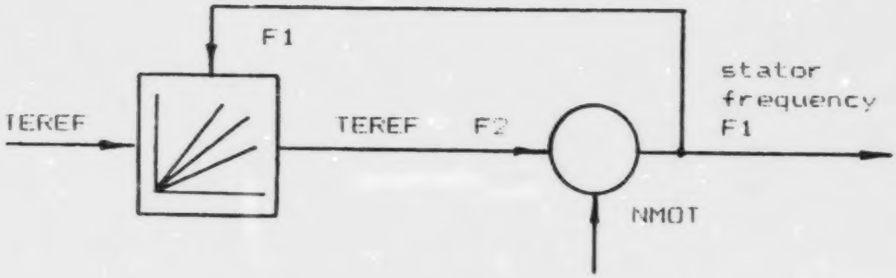
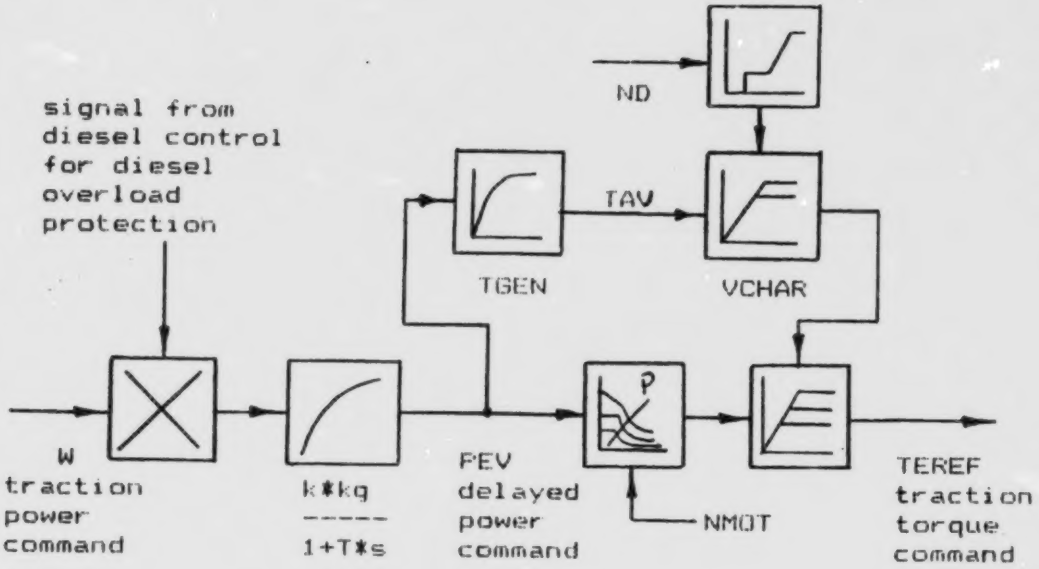

F1
stator
frequency
F1
TEREF
TEREF
F2
NMOT

Fig. 3.11 Inverter Simulation Block Diagram

The traction torque command control has the task to derive from the traction power command a traction torque command TEREF for the electric drive shown in fig. 3.11. The traction power command is provided by the locomotive controller.
The traction torque command is derived such, that diesel, inverter and electric motors are protected against overload and the desired traction specifications of the locomotive achieved. The simulation is shown in fig. 3.12.


signal from
diesel control
for diesel
overload
protection
ND
TAV
TGEN
VCHAR
P
W
traction
power
command
k*kq

1+T*s
PEV
delayed
power
command
NMOT
TEREF
traction
torque
command

Fig. 3.12 Traction Simulation Block Diagram

The protection of the diesel against overload is achieved by a multiplier in the input channel of the power command. This multiplier has normally the factor one. The factor is reduced by the diesel control, if the diesel is overloaded. This reduces the traction power command to the electric drive and unloads therefore the diesel.

The traction power command passes a filter to limit the rate of change to protect the inverter control. The traction torque is derived from the power command by the traction diagram. In steady state the power command has only discrete values, but during transients and if the diesel overload channel is active the power command value is no longer a discrete variable. Therefore the traction diagram has to provide values in the whole torque speed plane and not only at discrete power values.

Depending on the generator the dc link voltage at lower diesel speed may not be sufficient to support the commutation of the full motor current in the inverter. This property is modelled as a torque command limit depending on the diesel speed.

3.3.2 Quantities of the electric drive model

The per unit speed for the motors is chosen to be 3300 rpm. To have an integer scaling factor between the diesel model and the electric drive the per unit power for the electric drive is chosen to be four times the per unit power of the diesel or 14,666.7 kW. The resulting per unit torque for the electric drive is then 42,441 Nm. The peak torque of 4 × 9780 Nm is then 0.922 in per unit values. The rated torque in the traction diagram is 0.825 or 4 × 8750 Nm. The per unit value for the rated speed results for the rated power in notch eight and the rated torque. With the rated torque of 0.814 from table 3.1. scaled by 0.25 the rated speed of the motors is 0.247. The maximum torque values for each notch are evenly spaced between zero and the maximum torque. The power, torque and speed values are shown in table 3.2 and the traction diagram is shown in fig 3.13.

Traction Diagram Data

throttle notch	PDT diesel traction power p.u.	electric drive traction power p.u.	maximum traction torque p.u.	speed p.u. at end of maximum torque operation
1	0.01	0.003	0.115	.030
2	0.028	0.007	0.231	.030
3	0.064	0.016	0.346	.050
4	0.113	0.028	0.461	0.060
5	0.209	0.052	0.576	0.090
6	0.361	0.090	0.692	0.130
7	0.541	0.135	0.807	0.170
8	0.814	0.204	0.922/0.825	0.220/0.250

Table 3.2

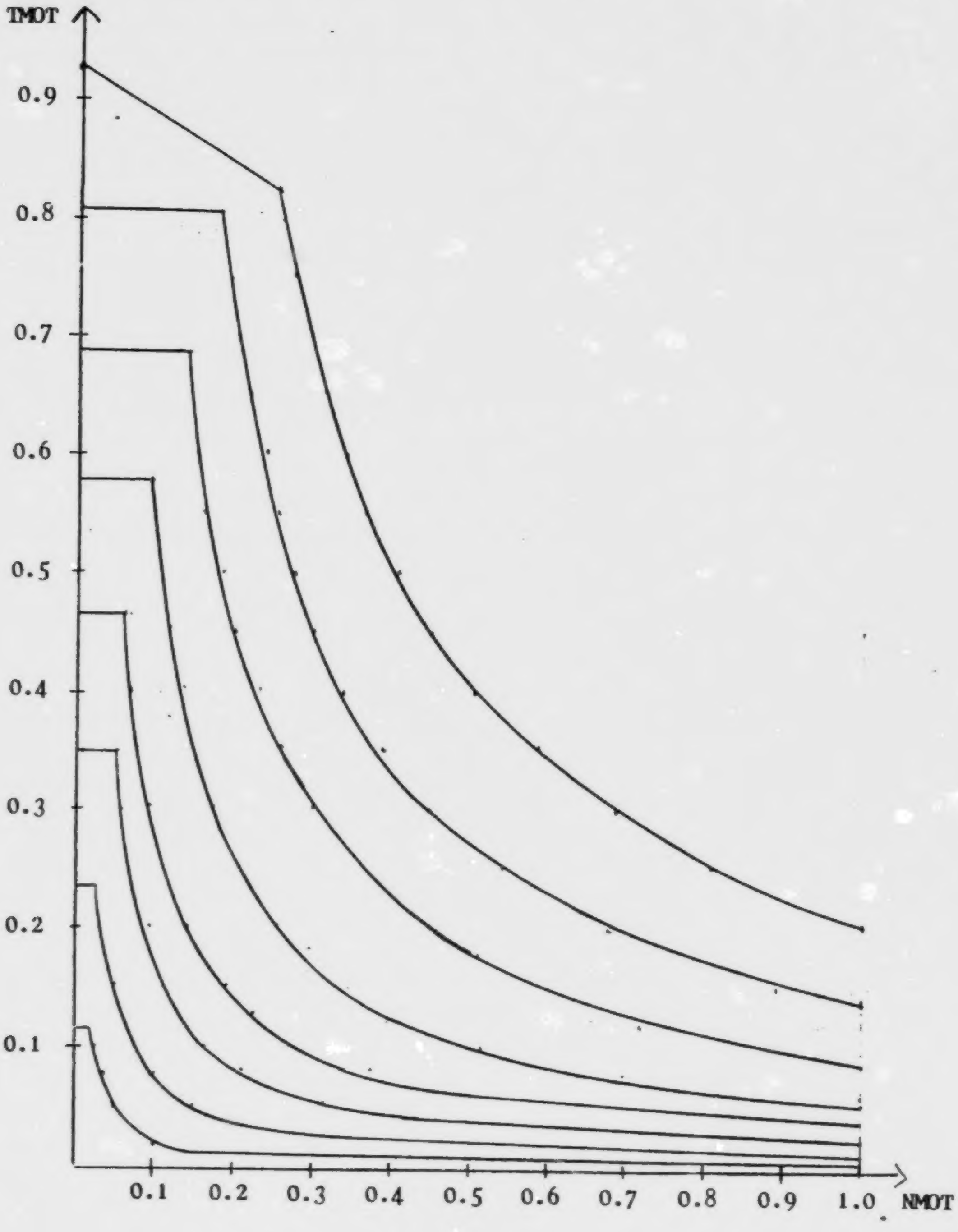


Fig. 3.13 Traction Diagram

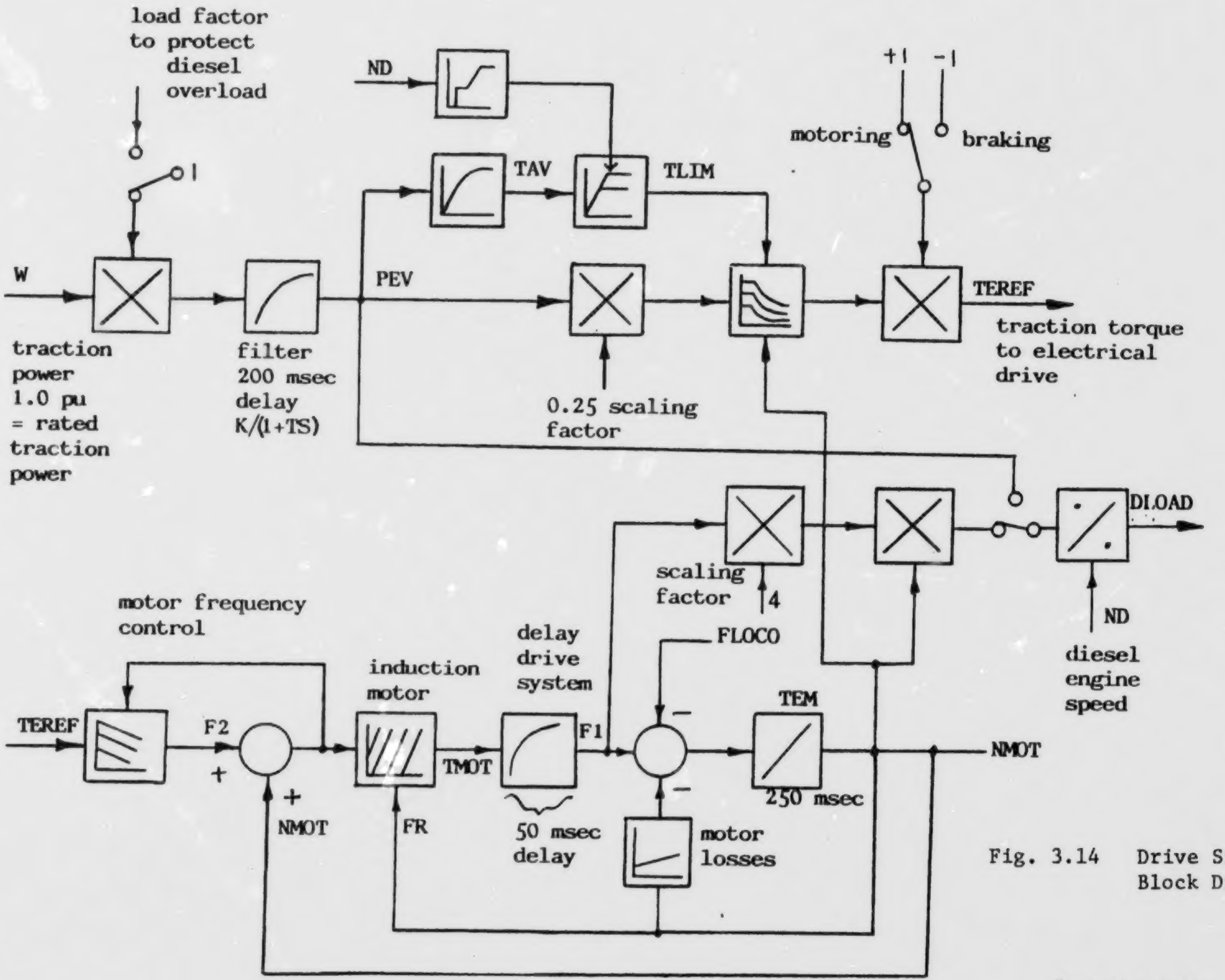

load factor to protect diesel overload
1
ND
TAV
TLIM
+1
-1
motoring
braking
W
PEV
TEREF
traction torque to electrical drive
traction power 1.0 pu = rated traction power
filter 200 msec delay K/(1+TS)
0.25 scaling factor
DLOAD
scaling factor
4
ND
diesel engine speed
motor frequency control
FLOCO
induction motor
delay drive system
TEM
TEREF
F2
+
+
NMOT
TMOT
FR
F1
–
–
50 msec delay
motor losses
250 msec
NMOT
electric drive simulation

Fig. 3.14 Drive Simulation Block Diagram

Table 3.1 and the traction diagram provide the quantities for the torque command control. The influence of the dc link voltage on the torque is modelled such, that the torque limit rises from 0.6 to 1.0 per unit torque between 400 rpm and 800 rpm diesel speed and is zero below the minimum diesel speed. The delay in the torque command control is modelled to 200 msec.

The inverter is modelled with a delay of 50 msec. This delay includes the electric transients of the motors. The torque to rotor frequency characteristic in the inverter is modelled as

$$f_2 = 0.03 \times torque \text{ for } f_1 < 0.4$$

$$f_2 = 0.03 \times (f1/0.4) \times (f1/0.4) \times torque \text{ for } f_1 > 0.4$$

with f_2 *= rotor frequency*

The rotor frequency to torque characteristic in the motor is the inverse of the inverter characteristic. The motor losses are modelled to *LMOT* = 0.005 × (1 + *NMOT*). The inertia of the motors is modelled by the integrator time constant of 1.2 seconds. This results from the calculation, that the rated torque of 8750 Nm accelerates the inertia of 25.6 Wsec3 to the per unit speed of 3300 rpm in 1 second. The complete block diagram of the simulation of the electric drive with the parameters is shown in fig. 3.14.

3.3.3 The program ELOCO and examples of the torque command control simulation

The electric motors and the inverter are simulated in the program module MLOCO, as they require a faster sample rate than the torque command control. MLOCO is described in section 3.4.3. The program module ELOCO contains the torque command control.

The program module ELOCO is executed by the program LOCO. The module ELOCO consists of several procedures or subroutines. A short description of these procedures and the listing of the program module is provided in the Appendix A2.
The block diagram of the simulation was shown in fig.3.14. The following fig. 3.15 to 3.17 demonstrate some aspects of the torque command control and verify the function of the real time simulation of the torque command control.
The core of the torque command control is the traction diagram fig.3.13. To demonstrate its implementation a constant power command W from the MICAS is set and the speed of the electric motors is allowed to rise slowly and nearly linearly from zero. The torque command TEREF follows then one of the characteristics of the traction diagram.

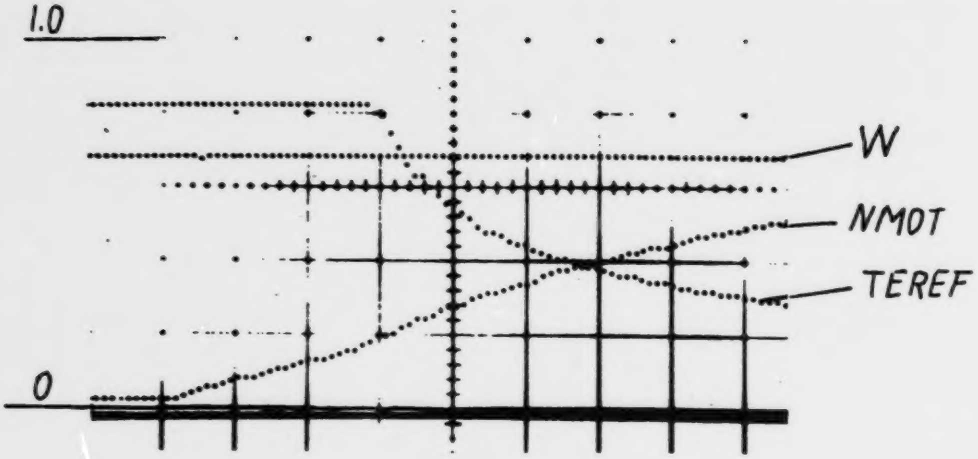


Fig. 3.15 Torque Command Simulation, Notch 7

In fig. 3.15 the locomotive operates in notch position seven, which is shown by the constant power command value W of 0.66 per unit. The speed rises from zero to about 0.5 per unit. The torque command stays at the maximum of 0.81 per unit until the operation with constant power is reached. The constant power operation is reached at 0.2 per unit speed. At 0.5 per unit speed the torque is 0.28 per unit.

Figure 3.16 shows the same variables as fig. 3.15 for an operation in notch eight.

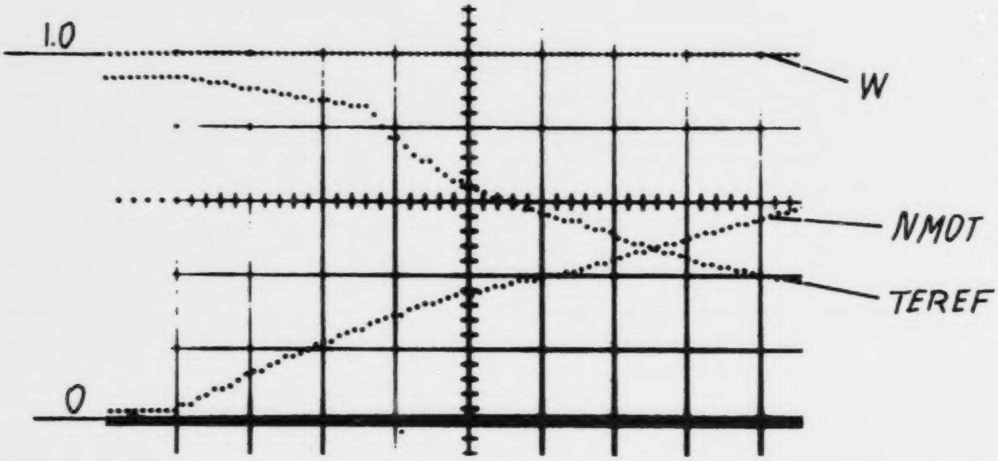


Fig. 3.16 Torque Command Simulation, Notch 8

It is clear to see, that the maximum torque in notch eight is speed dependent. The values of both figures confirm that the traction diagram of fig. 3.13 is implemented in the simulation.
That the command controller provides intermediate values in transients, which are between the steady state curves of the traction diagram is demonstrated in fig. 3.17. It also shows, that the MICAS controller provides the power command with

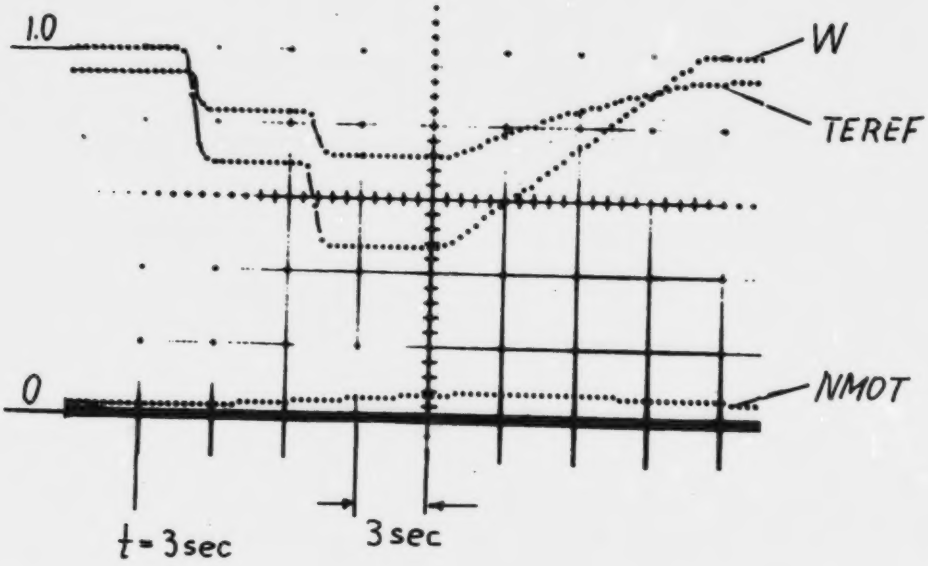


Fig. 3.17 Torque Command Simulation, Intermediate

a limited rate of change of 1.0 per unit per 20 seconds for the rising signal and with 1.0 per unit per 2 seconds for the falling signal. The figure shows the MICAS power command W, the command torque TEREF and the motor speed NMOT. The time scale is set to 3 sec per division.
The event shown is operation in notch eight at t = 0, with W = 1.0 per unit and TEREF = 0.92. At t = 5 sec the notch at the locomotive console is set to seven and at t = 10 sec it is reduced further to notch six. The transients are short and the command torque follows the power command dropping from 0.92 to 0.81 and 0.69 per unit torque. At t = 15 sec the throttle is set to notch eight again and the power command rises with its limited rate. The torque command rises continuously as well.

3.4 Simulation of the traction and the train

3.4.1 System description and simulation of the traction and the train

The traction is described by a characteristic providing the traction force as a function of the slip between wheel and rail. The slip is the difference between the speed of the wheels or motors and the true over ground locomotive speed.

Two mechanisms are provided to simulate a change of traction conditions. Up to three different sets of characteristics are stored in the computer and can be exchanged immediately during the simulation.
To change the quantities of a selected characteristic a multiplier is provided in the input and the output of the block simulating the characteristic. The multipier factors TRGTI and TRGTO are controlled during the simulation from the keyboard through the interactive simulation system. This allows to change the amplitude and slope of a given characteristic. Fig. 3.18a-c demonstrates these changes.

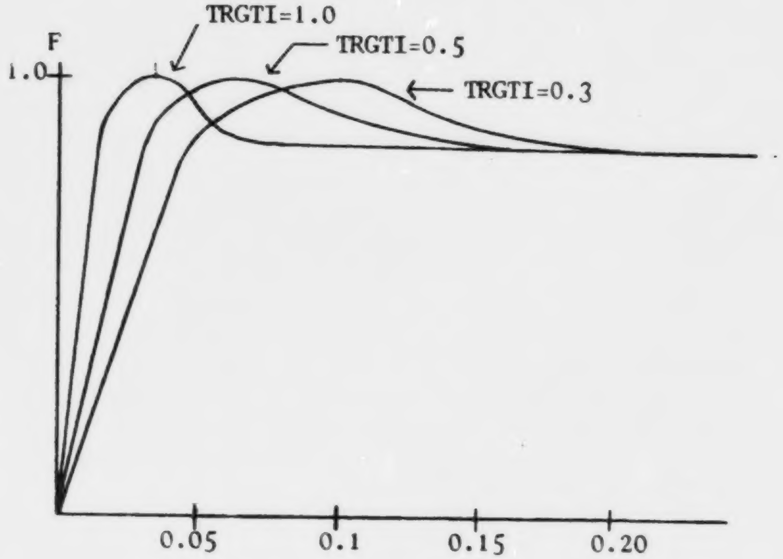


Fig. 3.18a Illustration of Characteristic Changes

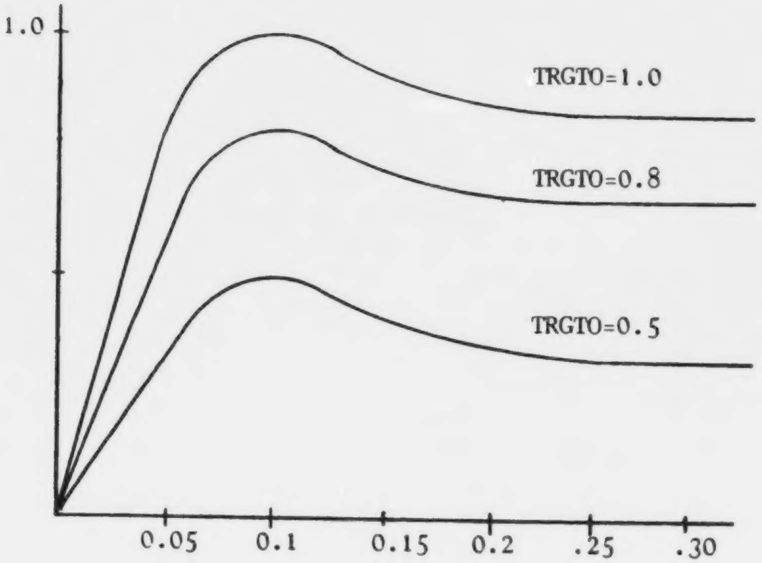


Fig. 3.18b Illustration of Characteristic Changes

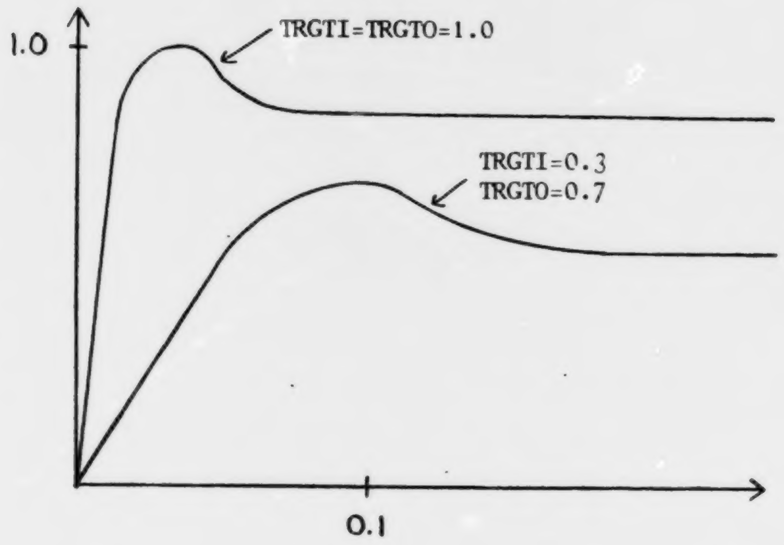


Fig. 3.18c Illustration of Characteristic Changes

To achieve a smoother transition than it is possible by switching characteristics and to control the transition time the change of the multiplier factors is controlled. Whenever a new value is set for TRGTI or TRTGO or both the transient to the new values is started by a key board command and takes a preselected defined time. The change from the old to the new value is a linear function of time. The structure of the traction and train simulation including some blocks of the motor simulation fig. 3.10 is shown in fig. 3.19.

The locomotive and the train are simulated as a single mass system and a load force. This simple model is sufficient for the studies at the locomotive. The load has a speed dependent part and a constant part. The constant part represents the force driving on a grade. The speed dependent part can be any function of the speed. For the project it was modelled as a linear function of speed, as this was sufficient for the problems studied.

A close examination of the structure in fig. 3.19 shows that the feedback of the traction force FLOCO to the summing point with the motor torque TMOT results in a transfer function with a time constant depending on the operating point on the traction force characteristic.
The fig. 3.20 shows the closed loop transfer function. The feedback of the motor losses is negligible. The locomotive speed NLOCO has no influence, as it changes so slowly, that it can be considered constant during the transients of the traction force. The constant kT depends on the operating point on the traction characteristic.

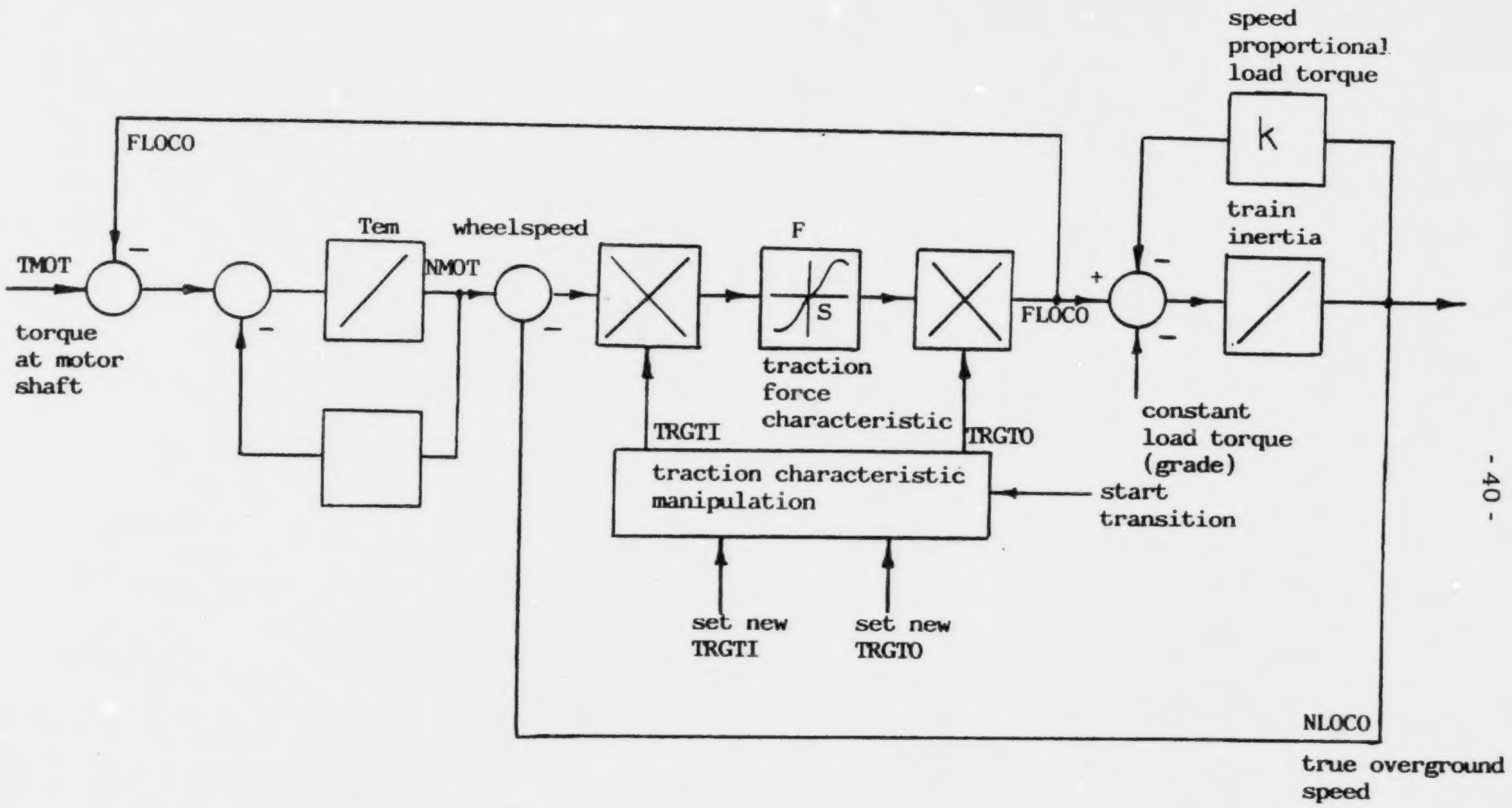

speed
proportional
load torque
k
FLOCO
TMOT
Tem
wheelspeed
NMOT
F
S
train
inertia
torque
at motor
shaft
traction
force
characteristic
FLOCO
+
constant
load torque
(grade)
TRGTI
TRGTO
traction characteristic
manipulation
start
transition
set new
TRGTI
set new
TRGTO
NLOCO
true overground
speed

Fig. 3.19 Traction & Train Simulation Diagram

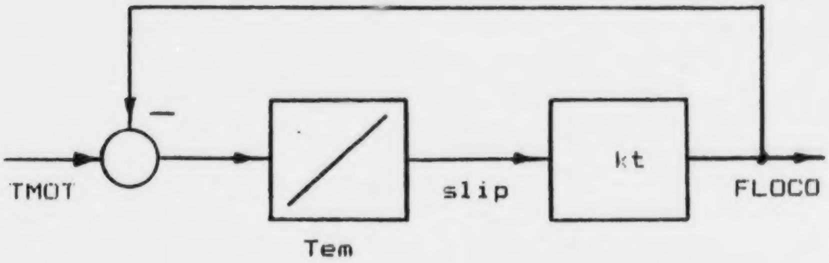


Fig. 3.20 Closed Loop Transfer Function

The resulting transfer function is

$$FLOCO/TMOT = 1/(1 + Tem/kT \times s)$$

As the traction force characteristic is a nonlinear function kT can be understood as the differential gain at the operating point. If the traction is good kT is in the order of 50 to 100 or the transfer function time constant is 50 to 100 times smaller than Tem. If slipping starts, kT becomes an order smaller.
The result is that the traction force build up and the increase of the slip at small values is considerable faster than the time constant of the electric drive Tem.

3.4.2 Quantities of the train and traction simulation

Three different traction force characteristics were used for the simulation. Two of them are of the type with maximum as shown in fig. 3.22 and one is of the type shown in fig.3.21. The parameter values for the characteristics as specified for the program CURVEGEN are given in table 3.4.

The train model has an integrator time constant of $n \times 10$sec. The variable n is selectable between 1 and 9 and allows to vary the train inertia. The train load is modelled as $k1 + 0.22 \times NLOCO$. The factor k1 can be changed at a potentiometer continuously between zero and one per unit. This variable preloading, which simulates operation at a grade, allows to simulate slip conditions at high traction torque.

The values of the multipliers TRGTI and TRGTO can be varied continuously between zero and one. The transition time between two settings is programmed to be $n \times 200msec$. The integer factor n is set from the keyboard and varies from 1 to 9.
There are up to three different traction characteristics stored in tables in the model at any time. It is possible to switch this characteristics immediately in the simulation.
The tables of the characteristics are contained in the program ASMCURVE. The tables and thereby the desired characteristics are generated by the program CURVEGEN. This program runs on the HP 64000 computer and is written in

HOST PASCAL. The program is interactive and asks for the input of different parameters to generate from them the desired characteristic. Two types of traction force over slip characteristics can be generated. The first type has no maximum and its specification is shown in fig. 3.21. The second type has a maximum. Its specification is shown in fig. 3.22. Both characteristics have an initial slope of 50, which results in two percent slip at per unit torque. The other parameters can be set by the program CURVEGEN. The parameters can be unsymmetrical for slipping and sliding. The figures 3.21 and 3.22 show two typical settings, with the parameters shown on the diagrams.

Parameters Traction Force Characteristics

	char. #1	#2	#3
slope	50	50	15
h+	0.75	0.8	0.75
r1+	0.1	0.05	0.1
r2+	n.a.	0.05	0.05
nsl+	n.a.	-100	-0.5
sl+	0.1	n.a. = 0	n.a. = 0
h-	0.75	0.8	0.75
r1-	0.1	0.05	0.1
r2-	n.a.	0.05	0.05
nsl-	n.a.	-100	-0.5
sl-	0.1	n.a. = 0	n.a. = 0

Table 3.4

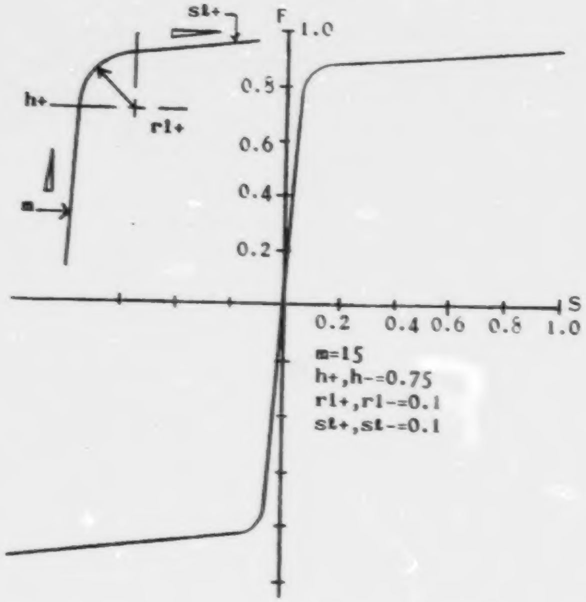


Fig. 3.21 Traction Force/Slip Characteristics - First Type

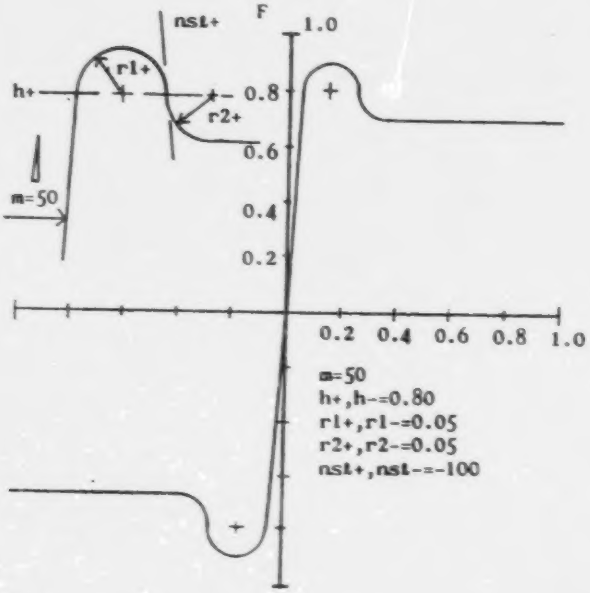


Fig. 3.22 Traction Force/Slip Characteristics - Second Type

3.4.3 The programs MLOCO,ASMCURVE,CURVEGEN,TLOCO and examples of the traction and train simulation

The program module MLOCO contains the simulation of the electric motors, the inverter and the traction force transmission between wheel and rail. The actual characteristic of the traction force as a function of the slip between wheel and rail is in the program module ASMCURVE. This module can contain up to three different characteristics stored in tables. ASMCURVE is used by the program module MLOCO.

The program module ASMCURVE including all tables is generated automatically by the program CURVEGEN to avoid a manual calculation and typing of the long traction force tables in ASMCURVE. This is done on the development system. The program module ASMCURVE is then linked with the other programs and down-loaded to the VME computer.

The program TLOCO contains the simulation of the train.

A short description of all the procedures of the programs and the programs themselves are contained in the Appendix A2. The program CURVEGEN is contained in Appendix A4.

The following examples of the simulation with MLOCO demonstrate different traction force characteristics and show that the slipping process is a fast transient as discussed before.

To demonstrate the implemented traction force characteristics the locomotive is brought into a slipping operation. The slip is then reduced to a small value by reducing the train load. The available traction force FLOCO is recorded during this transient and allows one to judge the traction force as function of the slip.

Two such events are shown in fig. 3.23.

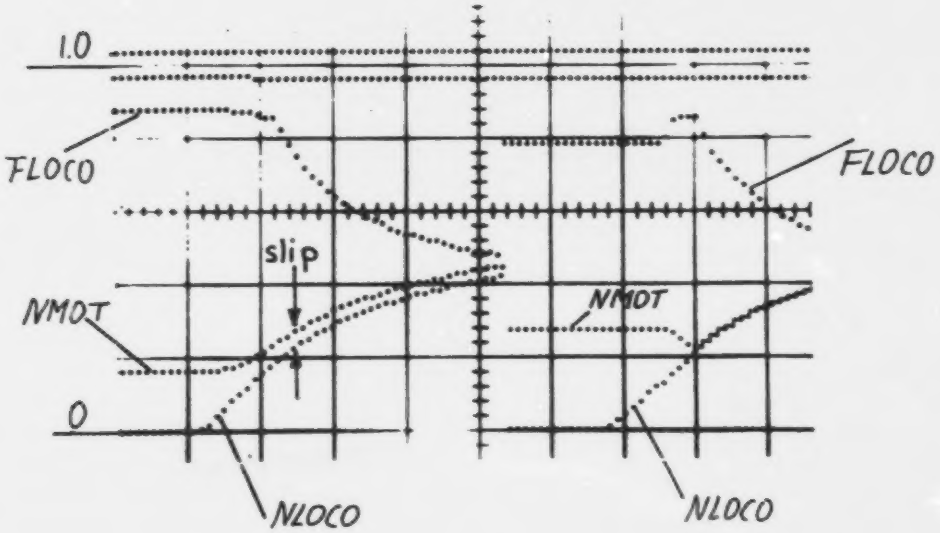


Fig. 3.23 Example of Slip Event

The first even represents the third characteristic used in the simulation the second event represents the second characteristic used. Starting at the left the bottom trace is the locomotive speed NLOCO. The second trace is the motor speed NMOT. The difference between NLOCO and NMOT is the slip. The third trace from the bottom is the traction force FLOCO. The two top traces are only registered for control purposes.
In the first event the slip falls from about 0.16 to about 0.04. The traction force falls monotonously as programmed for the third characteristic. Furthermore the characteristic is soft, as at half the per unit traction force the slip is still about 0.04.
In the second event the slip falls from 0.28 to nearly zero. The traction force shows clearly the maximum at lower slip values and the high force at very low slip values, as programmed for this characteristic.

The fig. 3.24 demonstrates a slipping transient with the traction force characteristic two. It shows at the bottom again the locomotive speed NLOCO and the motor speed NMOT. The third trace is the traction force FLOCO and at the top is the diesel engine speed for control. The time scale is set to 2 sec per division.

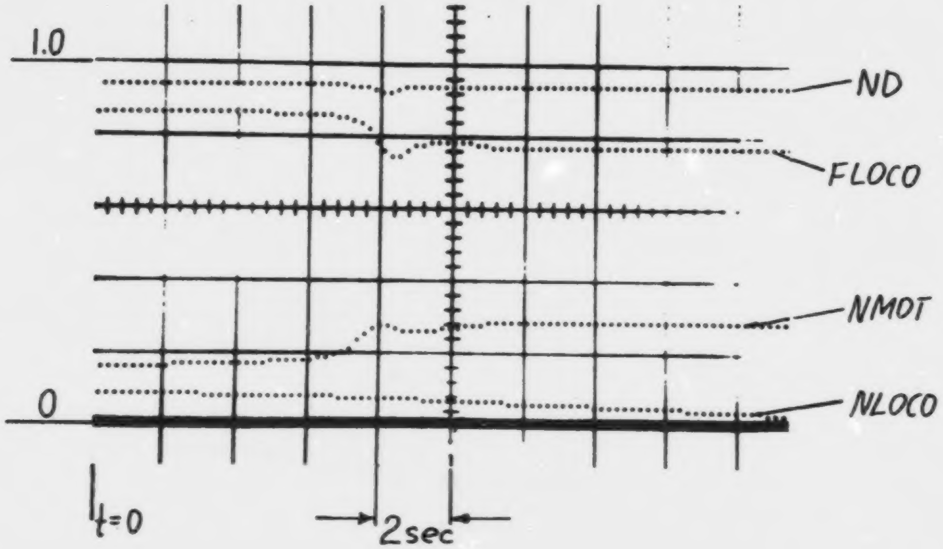


Fig. 3.24 Example of Slipping Transient

At t = 0 the locomotive is operating at the maximum of the traction force curve at a slip of 0.08 and a traction force of 0.85 in throttle position eight. As the available motor torque is marginally higher, the slip increases slowly until it reaches the negative slope of the traction force characteristic after the point of maximum force. Then the traction force falls to the lower value and the now larger difference between motor torque TMOT and falling traction force FLOCO accelerates the wheels in short time to higher slip values.
The sudden increase in slip occurs in about 0.7 sec and demonstrates the high

dynamic of the slipping process. The differential time constant of the transfer function as defined in 3.4.1 changes from a large value in the maximum of the traction force to a small value during the fast slip change, as the value of the feedback constant kT changes dramatically.

Another slip transient is shown in fig. 3.25. It shows again the locomotive speed NLOCO and the motor speed NMOT at the bottom. The third trace is the motor torque TMOT and at the top is the diesel engine speed. The time scale is set to 0.25 sec per division.

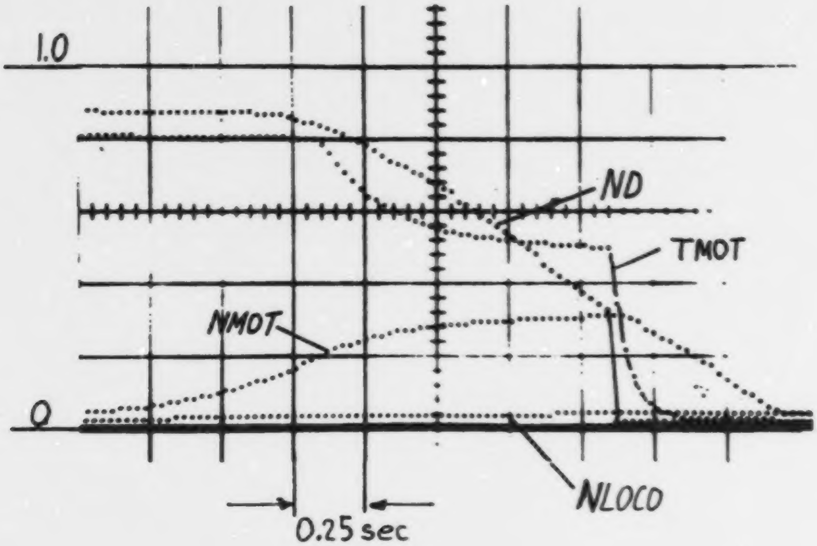


Fig. 3.25 Example of Slipping Transient

The locomotive operates in notch seven at a step grade with small speed and slip. The traction force characteristic is the curve two. At t = 0 the traction conditions are suddenly changed, by reducing the traction force multiplier factor TRGTO from 1.0 to 0.6 within 0.5 sec. As a result the maximum traction force, which can be transmitted reduces from 0.85 to 0.5. As the required force is 0.8 the locomotive will slip.

The figure shows, that the slip increases fast with about six percent per 0.25 seconds. If the motor speed reaches 0.2 the torque request is reduced, as the operation changes from constant torque to constant power request. This has a stabilizing effect, as the torque demand is reduced and therefore the rate of change of the slip is reduced. But at this stage the reduction is not enough to stop the slipping process.

Another effect of this slipping condition is of interest. If a slip is allowed to build up at a constant torque request, then the motor power and therefore the power required from the diesel engine rises as fast as the slip. The diesel engine due to the turbo charger is not able to accommodate this sudden power request, if it exceeds a few percent as it will be for severe slipping. The slipping condition shown requests 15 percent more power within 0.75 sec. The diesel engine

operating in notch seven can not provide the power within this short time and a severe overload condition occurs. As a result from the severe overload the diesel engine speed drops and the engine finally stalls.
The last effect indicates, that for a microslip control, which permits higher slip values, an overloading of the diesel engine might occur if no measures are provided to avoid it.

3.5 Real time simulation with the program LOCO

The program LOCO integrates all the simulation modules and the module IOLOCO. It executes these modules in the right order and controls the timing of the execution of the modules to provide real time execution. It sets up the timing and initializes the program variables. It generates a 1 Hz clock for the synchronization of the display. Two procedures support the data display by preparing the display data set selected by the keyboard and converting the data to the eight bit display format. The display data set is then ready to be transmitted by the procedure IRQ3 in IOLOCO on request from the DATDIS2 program. The principle structure of the program LOCO is shown in the flow chart in fig. 3.26. A module map, a short description of the procedures in the program LOCO and the program listing are in the Appendix A2.

The timing of the execution of the different modules in real time is governed by the theory of sampled data systems [1]. The sample rate must be set according to the bandwidth of the system. To determine the actual ratio between the sample time and the smallest closed loop system time constant the control design method and the resolution or word length of the computer have to considered.
For the project it was decided, that the sample time should be at least 6 to 10 times smaller then the smallest time constant. This selection permits to derive the algorithms by a bilinear transformation and the regulator design can be carried out using linear design methods, as the behavior of the linear system and the sampled data system are nearly the same [1].

On the other side the ratio of the sample time Ts and the largest time constant Tm should not be too large to avoid resolution problems due to the limited word length of the processor. In the algorithms the ratio Ts/Tm occurs as a factor together with the regulation error. As this error is small a ratio of 1/1000 or 1/10,000 for Ts/Tm as factor results very fast in values of the product, which are below the resolution of a sixteen bit processor. To avoid these the simulation uses different sample times in different modules and if this was not sufficient some algorithms are executed with 32 bit word length.

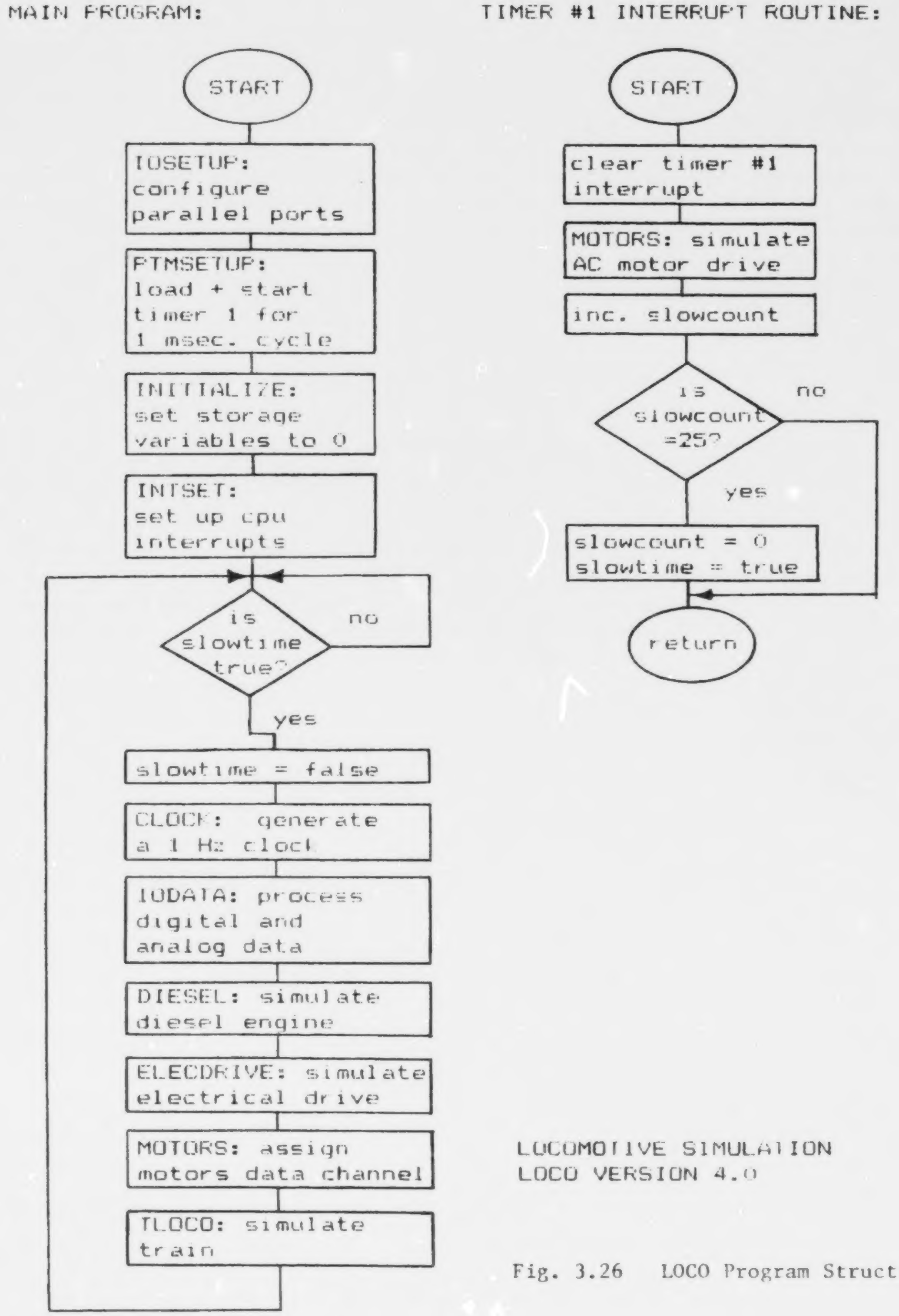

MAIN PROGRAM:
START
IOSETUP: configure parallel ports
PTMSETUP: load + start timer 1 for 1 msec. cycle
INITIALIZE: set storage variables to 0
INTSET: set up cpu interrupts
is slowtime true?
no
yes
slowtime = false
CLOCK: generate a 1 Hz clock
IODATA: process digital and analog data
DIESEL: simulate diesel engine
ELECDRIVE: simulate electrical drive
MOTORS: assign motors data channel
TLOCO: simulate train
TIMER #1 INTERRUPT ROUTINE:
START
clear timer #1 interrupt
MOTORS: simulate AC motor drive
inc. slowcount
is slowcount =25?
no
yes
slowcount = 0
slowtime = true
return
LOCOMOTIVE SIMULATION
LOCO VERSION 4.0

Fig. 3.26 LOCO Program Structure

To use different sample rates in a system is only possible, if the system is structured such, that a low pass filter limits the bandwidth at the link from a fast section to a slower section. The modules of the locomotive simulation with different sample rates were structured that this condition was fulfilled. The module MLOCO is executed every 1 msec. This time is controlled by the hardware timer under control of the program LOCO. The execution of this fast part is done in an interrupt routine, as shown in the flow chart of the program LOCO in fig. 3.26. The other modules are executed every 25 msec. This time interval is controlled by the program LOCO. The module TLOCO has an internal procedure, which slows down the execution of the control algorithm in this section even further, as the time constants in this section are much larger than in all the other modules.

3.6 The complete RIST system with the MICAS microprocessor control system

The complete RIST system and the MICAS system with all interfaces and the signals exchanged at the interfaces is shown in fig. 3.27. The MICAS system is operated for the studies with the software provided at its delivery. This software processed the logical signals and interlocks them. It changes the throttle setting in braking mode. It derives the power command signal W from the throttle position in motoring and from the braking effort potentiometer in braking mode. It limits the rate of change of the W-signal to 20 sec per unit for the rising signal and 2 sec per unit for the falling signal. The programmed slip control, based on a rate of change sensing of the locomotive axle speed was set at such a low sensitivity, that it was never activated and did not interfere with the studies. This setting was reprogrammed later.

The signal at the MICAS interface are the signal at the 15 V level. They are identical with the signal the processor will get from the general interface and provide to the inverter control logic.

The result of this algorithm is, that the microslip limit is reduced in each search step, if no definite gain in traction is sensed. This means if there is no gain, the system will always fall back after several steps to the smallest microslip limit SLOPMIN. This property of the algorithm also allows the search to stop before the maximum, as explained in fig. 5.4, by specifying a minimum gain as the condition not to step back.

The complete logical structure of the algorithm, including the maximum and minimum control of SLOP and the decision to search or not is shown in fig. 5.7. Each search step includes two passes through the structure in fig. 5.7. The time in between the passes is TSETTLE and the actual pass is governed by the logical variable new search NSEA, which is true at the begin of the search step.

5.5 Simulation of the proposed microslip control

The control system was simulated with the RIST system to verify the function of the proposed microslip control and the maximum traction force search algorithm. The program for the microslip control and the maximum search is incorporated in the program MLOCO. The program with a short description of the procedures is in the appendix A5, as stated earlier.

The fig. 5.8 shows the block diagram of the motor and inverter part

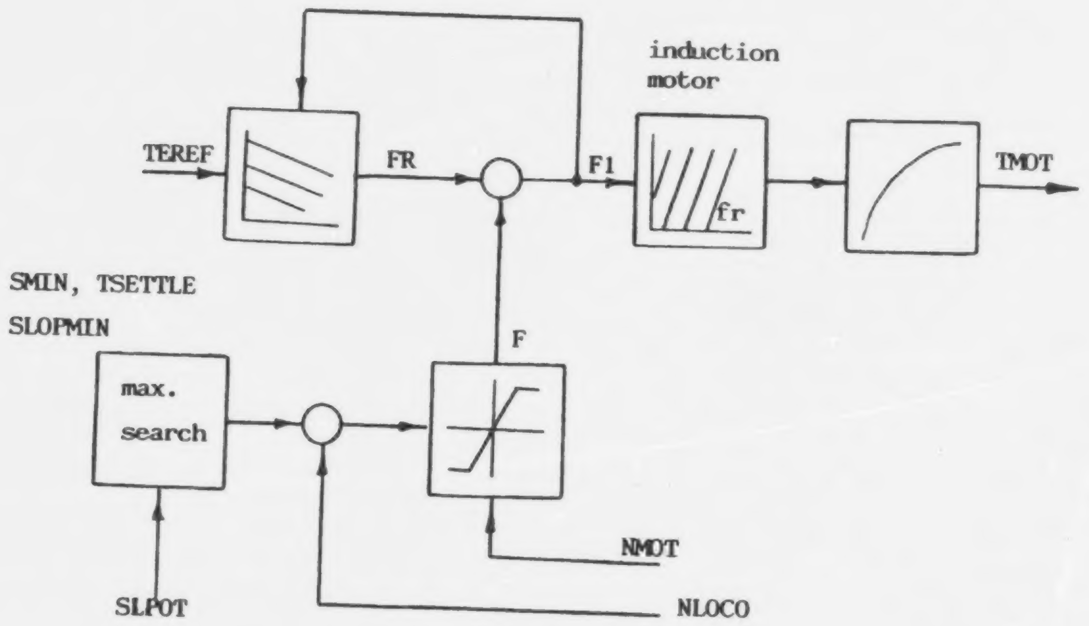


Fig. 5.8 Microslip Control Block Diagram

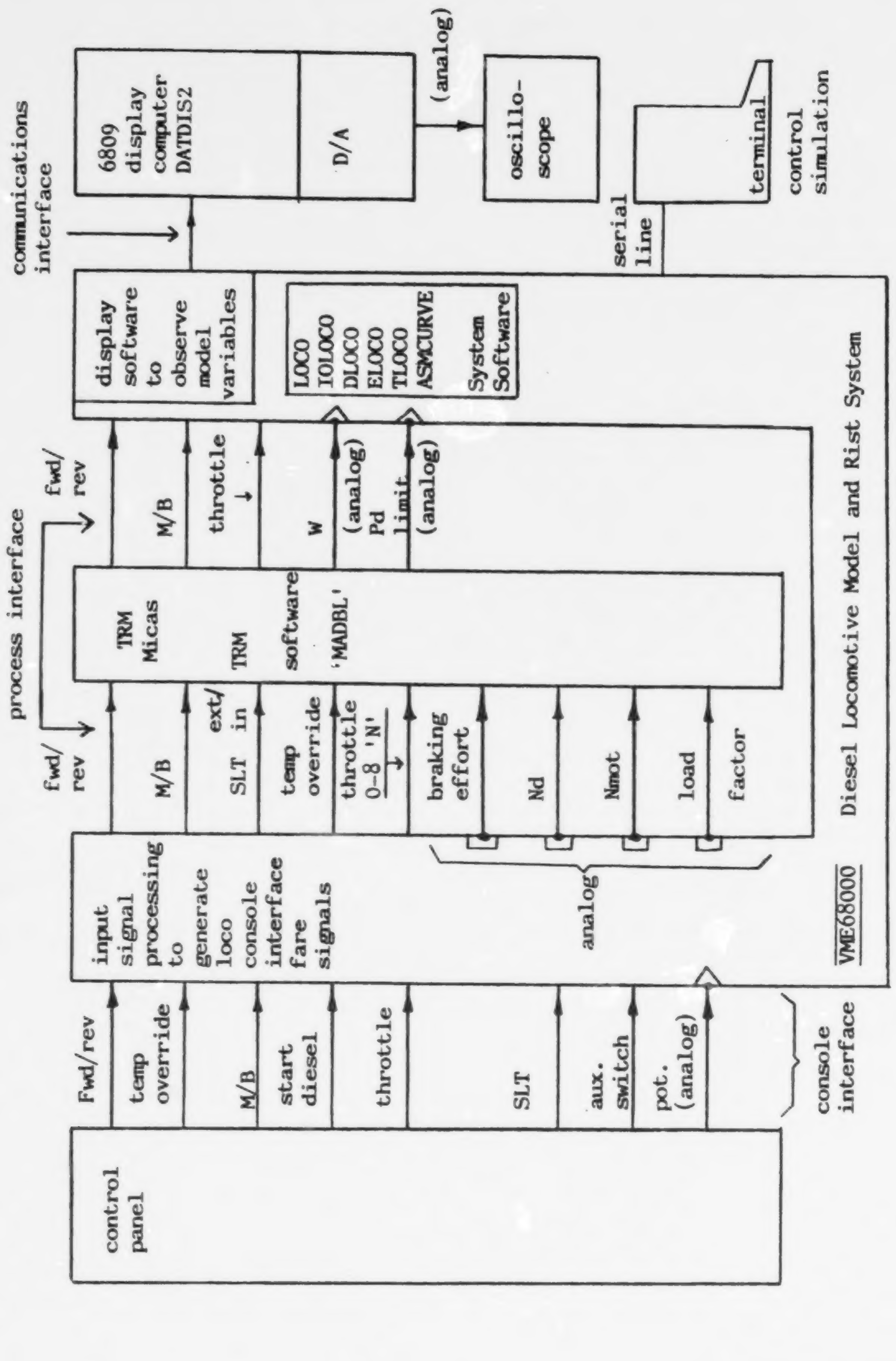

communications interface
6809 display computer DATDIS2
D/A
(analog)
oscillo-scope
terminal
control simulation
serial line
display software to observe model variables
LOCO
IOLOCO
DLOCO
ELOCO
TLOCO
ASMCURVE
System Software
Diesel Locomotive Model and Rist System
process interface
fwd/rev
M/B
throttle
W
(analog) Pd
limit (analog)
TRM Micas
TRM
software 'MADBL'
fwd/rev
M/B
ext/ SLT in
temp override
throttle 0-8 'N'
braking effort
Nd
Nmot
load
factor
input signal processing to generate loco console interface fare signals
analog
VME68000
Fwd/rev
temp override
M/B
start diesel
throttle
SLT
aux. switch
pot. (analog)
console interface
control panel

Fig. 3.27 Complete RIST & MICAS System

---

together with the new microslip control blocks. To use the locomotive simulation without the control algorithms, the maximum search can be activated or deactivated from the keyboard of the simulation control. The microslip control can be made inactive by setting SLPOT very high in the macroslip range.

In the following paragraphs some typical slip events are studied by simulation to test the proposed microslip control concept. The parameter selection for the simulation is discussed and results of slip transients observed on the display of the RIST system are presented.

5.5.1 Parameter selection for the microslip control study

The locomotive model as described in chapter 3. is used for the simulation. The set of traction force curves from table 3.4 is used together with changing factor TRGTO. In most tests the characteristic #2 was used with a factor TRGTO of 0.5, 0.6, 0.9 or 1.0. In one case the characteristic #3 with factor 1.0 was used. The fig. 5.9 shows the collection of traction force characteristics used for the microslip studies.

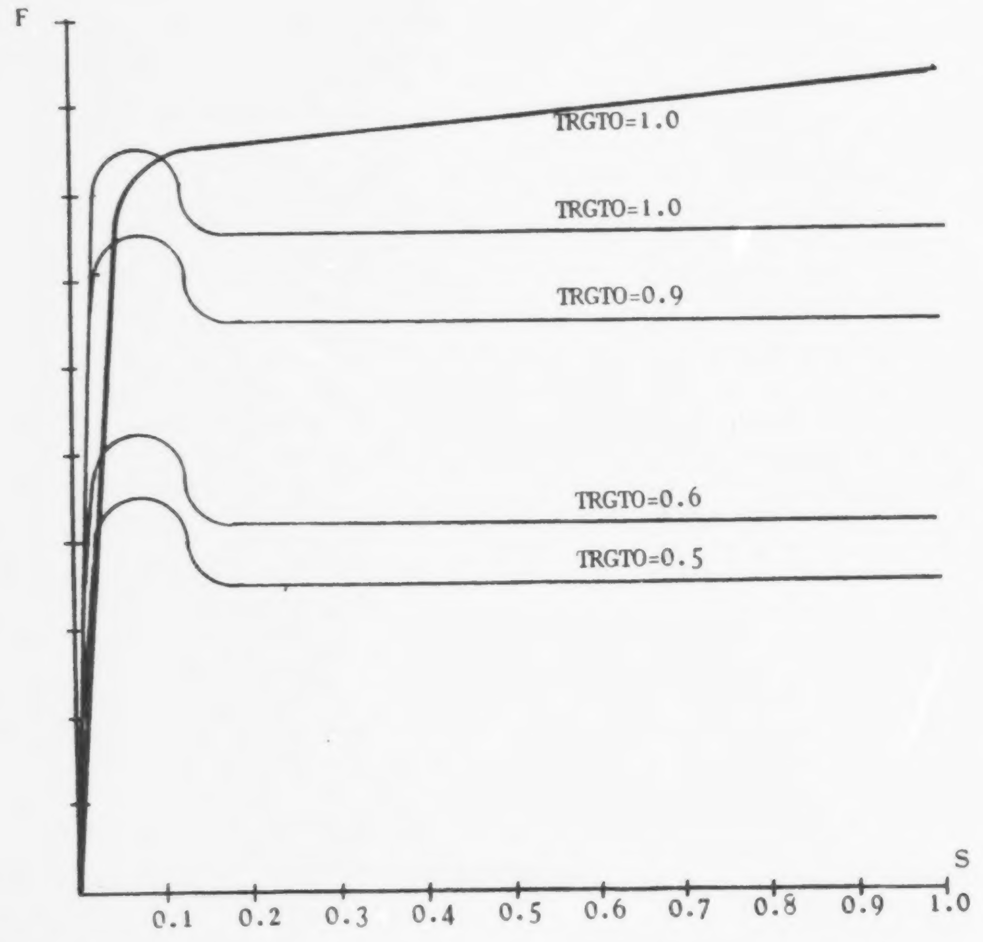


Fig. 5.9 Microslip Control - Traction Characteristics

4. System Control Studies

The locomotive system was thoroughly analyzed to set up the real time simulation system. After this it is possible to select the controls, which should be studied to be implemented with the MICAS microprocessor control system. Depending on the results the controls may be implemented. The control algorithm is designed and programmed and then tested on the RIST system to verify the expected performance.

Three major areas have been selected for the studies. The first is the traction control, the second is the data acquisition and data processing for on board monitoring and diagnostics and the third is the diesel engine and the locomotive speed control and their impact on fuel consumption.

The study of the traction control should develop a concept to control the slip of the wheels of the ac-dc-ac locomotive. The traction motors of the locomotive are ac induction motors with squirrel cage rotors. The slip of the wheels in driving and braking should be controlled under the assumption, that the true over ground speed of the locomotive can be measured. The control should not only limit the slip to avoid the range of macroslip conditions, but should control the amount of slip to operate the locomotive at the maximum available adhesion in the microslip range.
The developed concept should be compared with slip controls for diesel electric locomotives with dc traction motors.

The study of a monitor and diagnostic system should investigate how the MICAS microprocessor control system can be used for this general data acquisition and data processing task. A concept how to use the MICAS system for such a task should be developed.
The monitor and diagnostic system should be able to collect data during the operation of the locomotive and to generate from these data a performance record and information for maintenance. The system should, if possible, record fault events with a high resolution of the time, to generate reports for fault analysis and diagnostics. Finally the system should be flexible enough to be altered by the operating railway company, if the maintenance or diagnostic procedure is changed.

The study about diesel engine control investigates controls to improve fuel consumption. It is studied how variable diesel engine speed can contribute to economical fuel usage. Certain operational conditions like speed controlled locomotive operation and multiple unit traction are studied with the aim to propose control strategies for these operations, which minimize fuel consumption.

Beside the three major areas selected to be studied, and as a result of these studies, a control for overloaded protection of the diesel engine was carried out, as this control might be desirable.

The diesel engine overload protection control may be required, if an operation of high microslip values is desired. It can be of general use to protect the diesel engine against overload, and this control opens the possibility to utilize the full available diesel power, if this is desired.

The specified system controls will be discussed in the next chapters.

5. Traction Control

As a result from the system study of the ac-dc-ac diesel electric locomotive the following facts have to be taken into account for the design of a microslip control. In this section a proposal for the traction control is developed. First the system is described, then the specifications for the traction force control are defined. A control strategy is developed and its feasibility is studied on the RIST system by simulation. The implementation of the proposed control is discussed and finally a program for the MICAS microprocessor controller is written.

5.1 System description and analysis

As a result of the system study of the ac-dc-ac diesel electric locomotive the following facts have to be taken into account or considered for the design of a microslip control.

5.1.1 Traction force characteristic

The traction force characteristic although modelled as a function is not well defined. It is normally a wide band of values from hundreds of measurements [2],[3]. The same applies to the definition of a curve of maximum traction force, which can be transmitted. Furthermore these characteristic bands seem to be influenced by dynamic events coming from rail joints and curves. As a result all control algorithms assuming a well defined slip-traction curve will fail.

Keeping in mind, that the traction force characteristic describes such a band, the following general parameters defining these bands may be used to develop control strategies.
The traction force characteristic rises very steep in the beginning. It flattens then either with a monotonous decrease of the slop, or it reaches a maximum force, and falls with increasing slip again. The steep part of the characteristic occurs typically at slip values of less than one percent. A friction maximum may occur at slip values higher than three percent. It is not predictable which type of characteristic occurs and if there is a maximum it is not predictable where it occurs.
The traction force, which can be transmitted, is not the same for all axles. Particular for contaminated rails usually the traction force increases from the leading to the trailing axle.
The friction conditions can change within seconds, as the locomotive moves on to contaminated rails.

5.1.2 The ac drive

The ac drive are squirrel cage induction motors fed from a common bus. The bus supplies variable stator frequency and variable stator voltage. The voltage is controlled to supply the desired magnetic field in the motors. The frequency determines the speed.
The natural speed regulation of the induction machine of this size is very good. Typically it is in the range of one percent. The result is, that the speed difference between the axles is so small that no single axle slipping is possible.

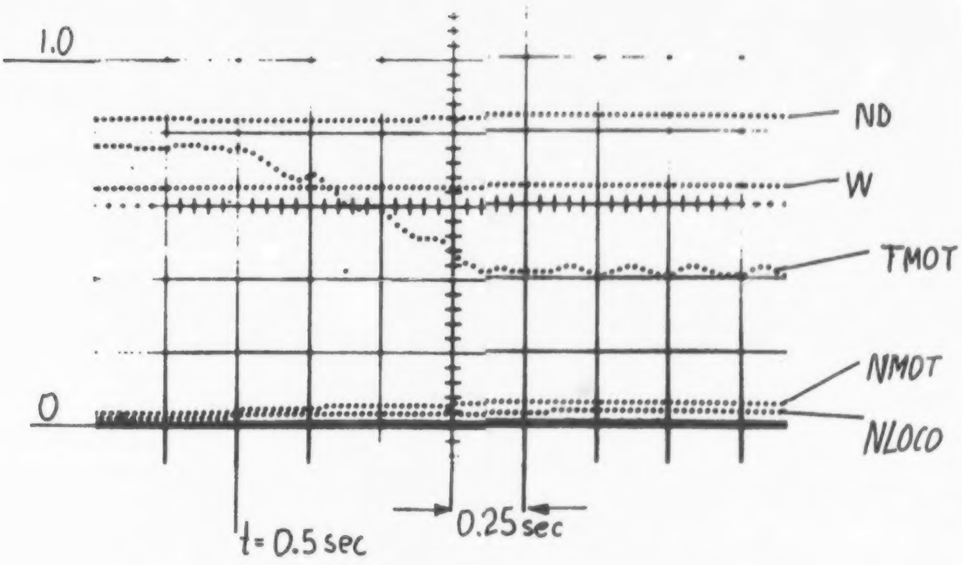


Fig. 5.11 Slip Limitation to Set Values

Figure 5.11 shows that sudden changes in slip conditions even of considerable amplitude are controlled fast by the proposed microslip control.

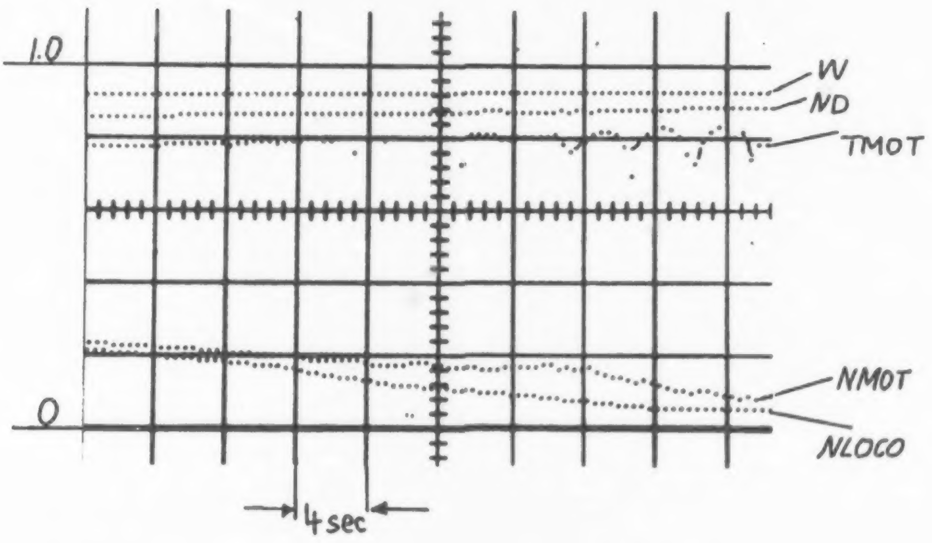


Fig. 5.12 Control of Changes in Slip Conditions

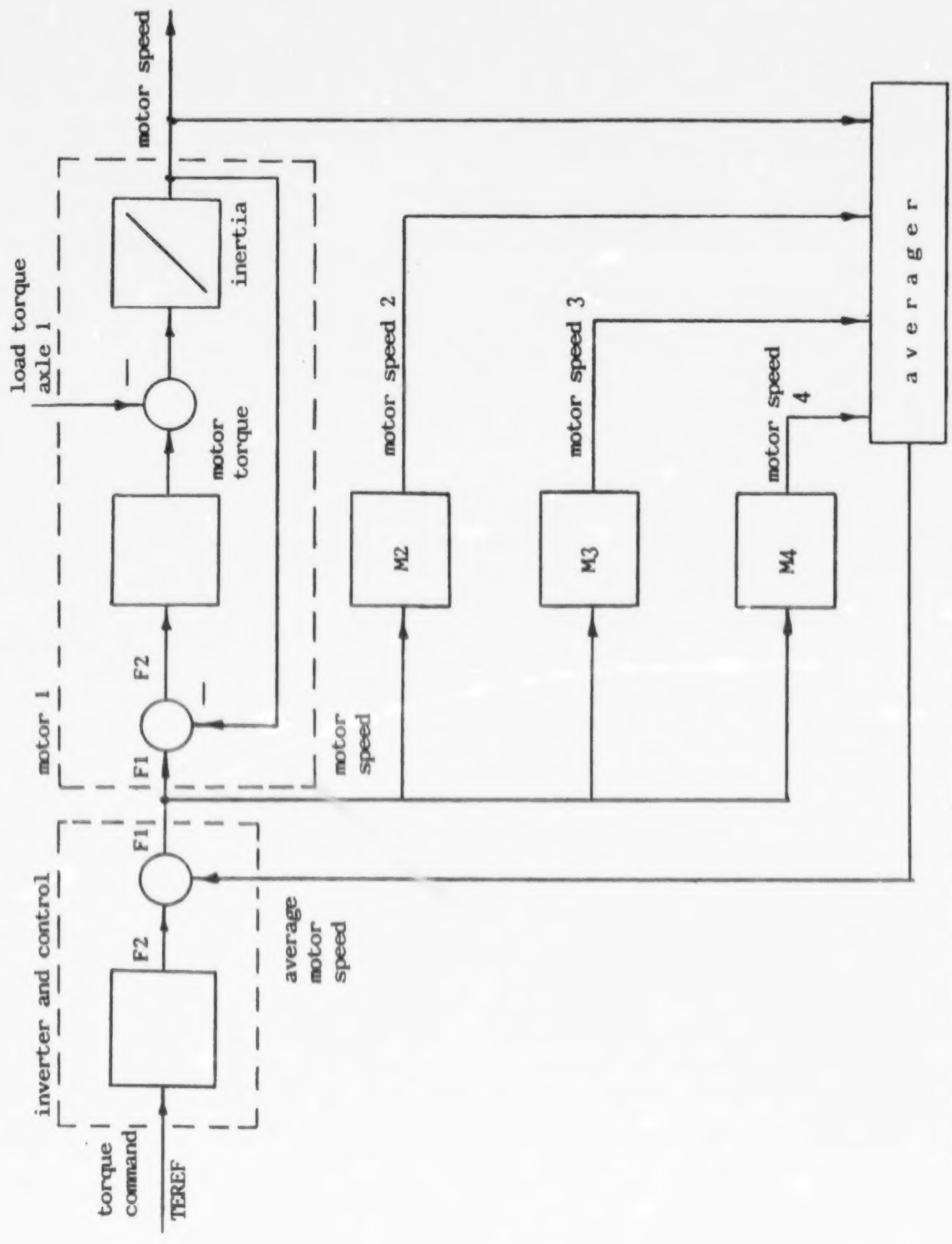

torque command
TEREF
inverter and control
F2
F1
average motor speed
motor 1
F1
F2
motor speed
motor torque
load torque axle 1
inertia
motor speed 1
M2
M3
M4
motor speed 2
motor speed 3
motor speed 4
a v e r a g e r

Fig. 5.1 Four-Motor Drive Structure

An event similar to fig. 5.12 was shown earlier in fig. 3.24. It showed the locomotive operating in notch 8. The traction characteristic is #2 with TRGTO = 1.0. The load is slightly higher then the maximum friction. Therefore the slip is slowly increasing as the locomotive speed drops. At t = 7 sec. the slip reaches the point of maximum friction the traction force drops and the slip increases suddenly to 30% as the microslip limit is set at 30%.

Figure 5.12 shows the same initial condition as Fig. 3.24, but this time the microslip control with maximum traction force search is activated. The figure shows the locomotive speed NLOCO,the motor speed NMOT, the motor torque TMOT, the diesel engine speed ND and the power command W. ND and W are shown for control purposes only. The time is 4 sec per division.

The locomotive operates as before in notch 8 with with the same traction force characteristic and is heavily loaded, so that the train speed drops. The slip builds up, but never passes the maximum friction, as the maximum traction force control maintains this operating point. At the torque TMOT, which is about 0.8, it is clear to see how the search algorithm oscillates around the maximum friction point.

This simulation shows, that the proposed control is able to stabilize the operation at the maximum friction point, as expected.

The next two figures show the capability of the search algorithm to search backwards.

Figure 5.13 shows the locomotive speed NLOCO,the motor speed NMOT, the motor torque TMOT, the diesel engine speed ND and the power command W. NLOCO is always zero. ND and W are shown for control purposes only. The time is 4 sec per division.

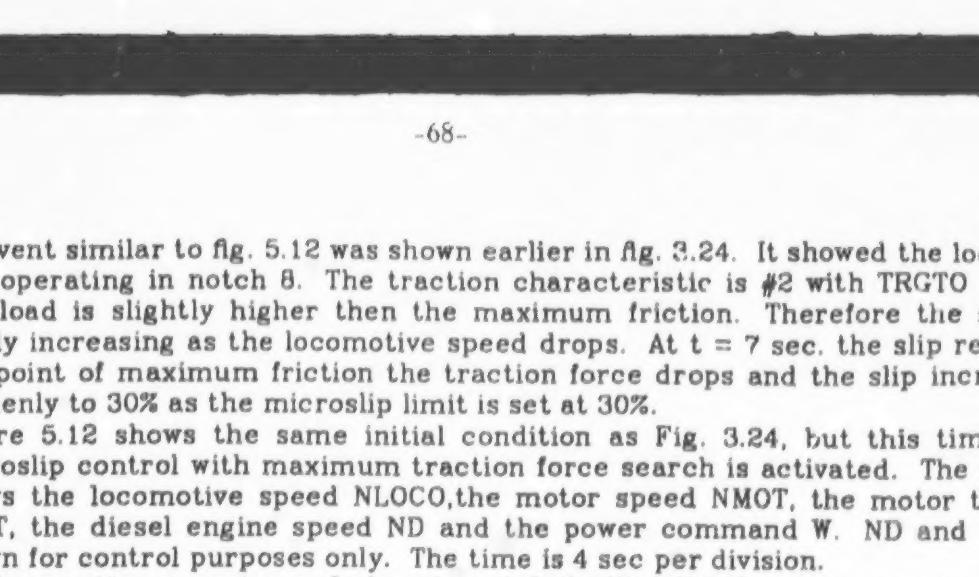


Fig. 5.13 Microslip Control - Backward Search

The locomotive operates in notch 8 with a heavy train (90 sec) on a steep grade.

The motor, with its supply the inverter, is configured as a torque controlled drive, as most transportation drives are. The principle structure of a four motor drive, neglecting electrical transients, is shown in fig. 5.1.

The torque command results in a rotor frequency command f2. The rotor frequency is added to the motor speed and the stator frequency f1 results. The motor torque is proportional to the rotor frequency. If the motor torque is equal to the load torque the motor speed is constant. If the motor torque is larger as the load torque the motor speed increases. If the motor speed increases, the stator frequency increases as well, through the speed feedback. Thus the rotor frequency is kept constant, which results in a constant motor torque, as it should be for a torque control.

As shown in fig. 5.1 all motors get the same frequency and this frequency is the sum of the rotor frequency command and the feedback of the average motor speed. The feedback of the average speed has the effect, that if one motor is unloaded due to bad friction conditions at the axle, the other motors take momentarily over its load. This means the locomotive does not loose power, if one axle has bad conditions but the other axles can support the required total traction force.

The control of the motor speed in this system is in principle unstable, as it is for all torque controlled drives. If the load torque is smaller than the commanded torque the speed increases infinitely. This will always happen, if a slipping condition occurs. The motor torque exceeds then the load torque, which is reduced by the reduced available maximum friction.

There are two principle possibilities to interrupt this run away condition. The first is to change the command torque TEREF to match the load torque and thus keep the speed constant. The second is to interrupt the feedback process temporarily by a limiter. In this case the stator frequency stops increasing, but the excessive torque still increases the motor speed, which in turn reduces the rotor frequency in the motor and lowers thus the torque. The process stablizes, when the motor torque matches the load torque.

5.1.3 Sensing of the slip condition

As the intention of the microslip control is to control the slip at certain values to utilize the maximum available friction, it is necessary to determine the actual slip value. A sensing of the rate of change of the motor speed only is not sufficient for this purpose. To determine the slip it is necessary to measure the true over ground speed of the locomotive. It is assumed, that this is possible either with a tachometer mounted at a non driven locomotive axle or with another device such as a radar sensor [4].

To accommodate different measurement systems it is assumed, that the true locomotive speed is available with an accuracy of 0.5 precent and that the measuring starts at 1.5 percent of the locomotive speed. The dynamic properties of the measuring system are assumed to be not worse then a first order delay with a time constant of 0.25 sec.

5.2 Specifications of the traction force control system

To illustrate the specifications fig. 5.2 is used. It shows the available

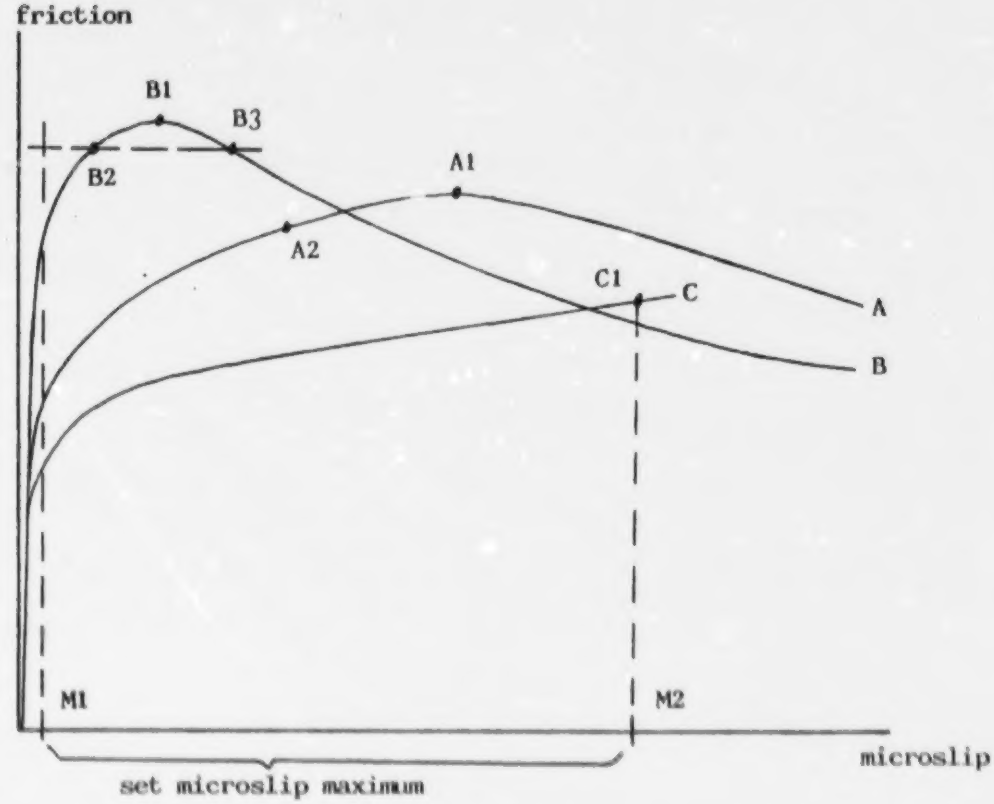


Fig. 5.2 Tractive Force Control Specifications

friction as function of the microslip. Three characteristics are shown to demonstrate the desired operation.

The control can be set to avoid slipping or limit it to very small values as conventional slip controls. Such a setting would be a microslip value M1 in fig. 5.2. The system would operate only between 0 and M1 slip an the characteristics A,B and C.
Continuous operation at increased slip or microslip should be possible for optimal utilization of limited friction under bad rail conditions. This would mean operation in the points A1,B1 or C1 in fig. 5.2.
The maximum of the microslip up to which the system is allowed to operate can be set. If set to M2 the system could operate between 0 and M2 slip in fig. 5.2, preferably it will operate in A1,B1 or C1. In the extreme case, if it is set very low i.e. M1 it approaches the case were slipping is avoided, as specified above.
If there is a maximum friction condition for a slip lower than the maximum

allowed microslip, the system should operate at a slip utilizing this maximum. For fig. 5.2 it means operation in A1 or B1.
The system should search for the maximum friction utilization within the given microslip range. It should search forwards and backwards. If, for example, the system operates in point A1 in fig. 5.2 and the friction condition changes to characteristic B, the system should go from the operating point A1 to B1.
Searching the maximum, i.e. B1, the system should stop the search close to the maximum on a positive slope at B2 rather than in B3. This provides near maximum traction at minimum possible slip to avoid unnecessary wheel and rail wear. In the same way it should stop in A2 rather than in A1 to get nearly the same force at half the slip.

Under bad rail conditions the system should maintain the traction force as high as possible without unnecessary long or deep reduction of the traction force.
The system must be able to accommodate fast and large changes in friction conditions and maintain a stable operation.

5.3 Control strategy

To control the slip it is proposed to limit the average motor speed feedback value. This takes advantage of the natural characteristic of the ac motor. It is expected that the system provides a good dynamic performance and reduces the traction force only as much as necessary not to exceed the set microslip limits.
Limiting the feedback of the average motor speed has the result, that a necessary torque reduction happens first at that motor, which increases its speed and makes the average speed exceeding the limit. But this is most likely the axle with the worst friction conditions.

The block diagram of the microslip control connected to the model of the locomotive is shown in fig. 5.3. The microslip limit command is added to the measured true locomotive speed and provides the upper limit for the feedback of the average motor speed NMOT. The slide limit subtracted from the locomotive speed controls the lower limit of the frequency F1.

As long as NMOT is as low as the limit the microslip can build up, or if normal friction condition prevail no increased slip occurs. As soon as the limit is reached, the frequency F1 stops rising and the motor torque is reduced according to the motor characteristic. This control limits the microslip to a value determined by the set microslip limit plus the error in the measurement of the locomotive speed plus the contribution DSLM through the regulation of the induction motor. As the regulation is very good, typically in the order of one percent for the full load, a torque reduction of 20 percent results in a value of 0.2 percent for DSLM, as an increase of the microslip. The microslip is limited no matter which type of traction force characteristic is effective. The limit varies slightly depending on the amount of torque reduction within the regulation of the motor characteristic.

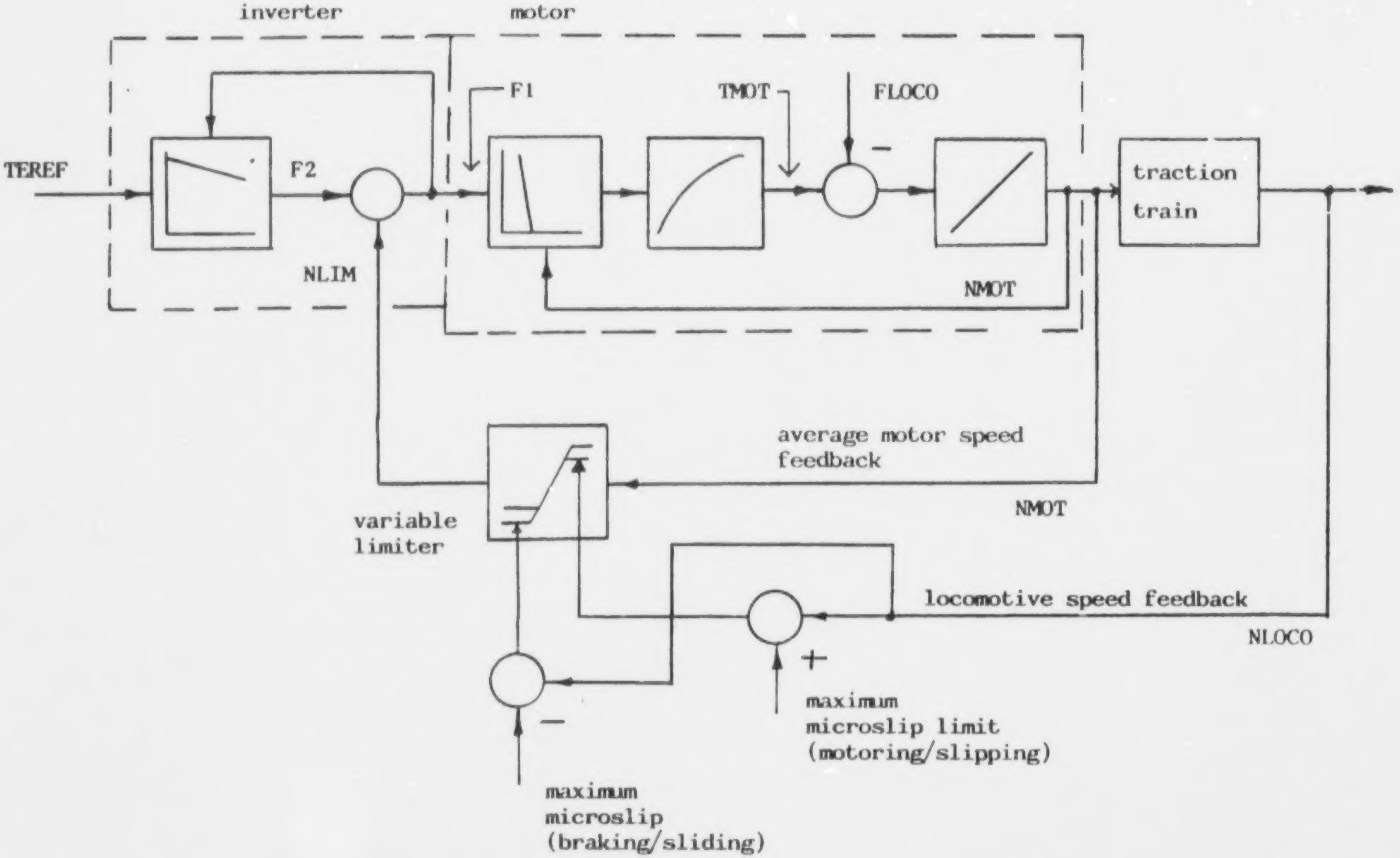

inverter
motor
TEREF
F2
NLIM
F1
TMOT
FLOCO
–
NMOT
traction
train
average motor speed
feedback
NMOT
variable
limiter
locomotive speed feedback
NLOCO
+
–
maximum
microslip limit
(motoring/slipping)
maximum
microslip
(braking/sliding)

Fig. 5.3 Microslip Control with Locomotive Model

Fig. 5.4 illustrates an example, where the microslip limit is 3%, the locomotive speed error +0.5% and the regulation of the motor for full load is 1%. The slip is limited within a very narrow band independent of the assumed traction force characteristic A,B and C.

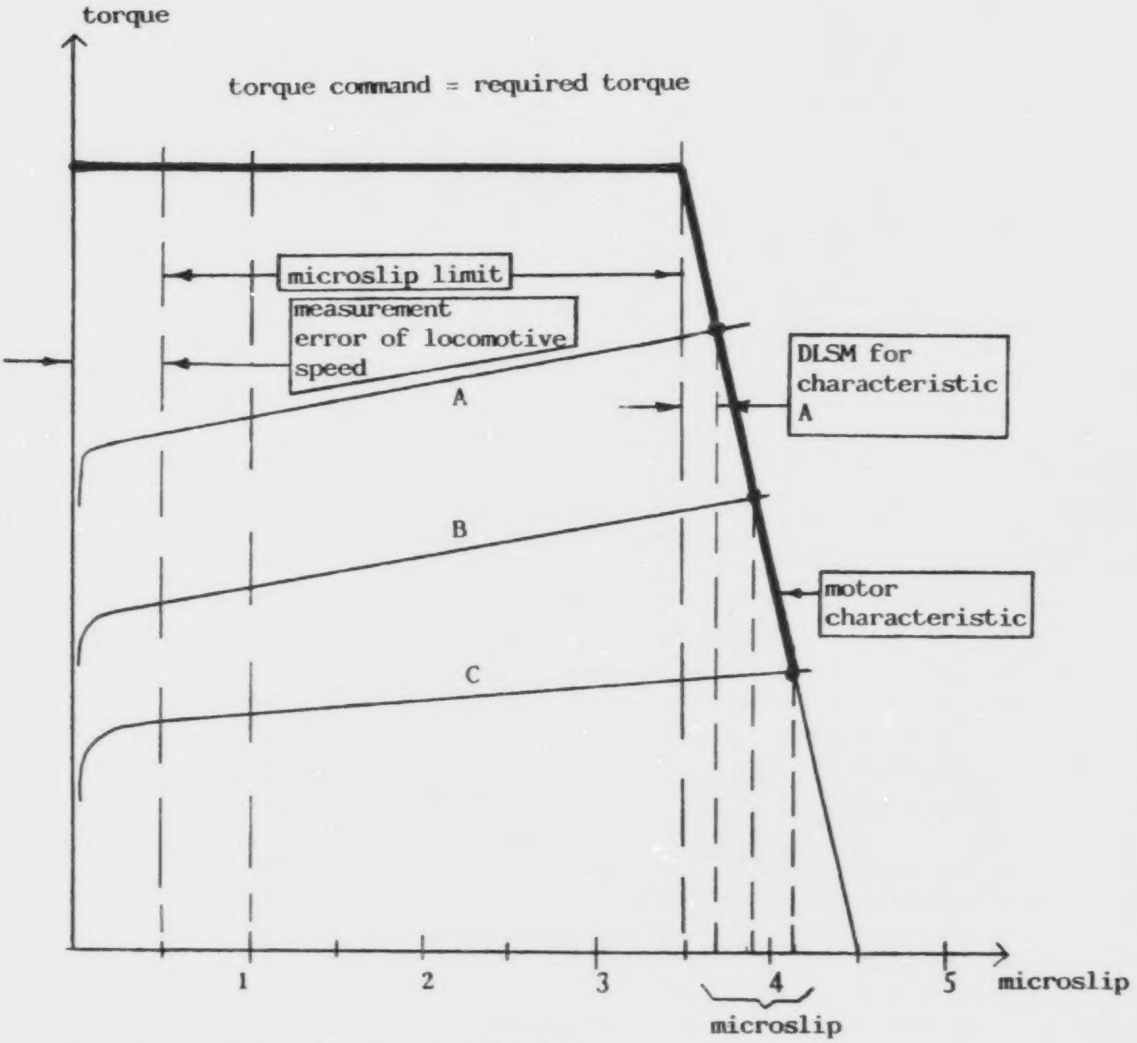

torque
torque command = required torque
microslip limit
measurement
error of locomotive
speed
A
DLSM for
characteristic
A
B
motor
characteristic
C
1
2
3
4
5
microslip
microslip

Fig. 5.4 Microslip Control Example

Figure 5.5 shows, that the electric drive characteristic with maximum microslip control is a horizontal line given by the requested torque and a near vertical line starting at the set maximum microslip SLOP. The near vertical line is the characteristic of the electric motors fed from a common bus.

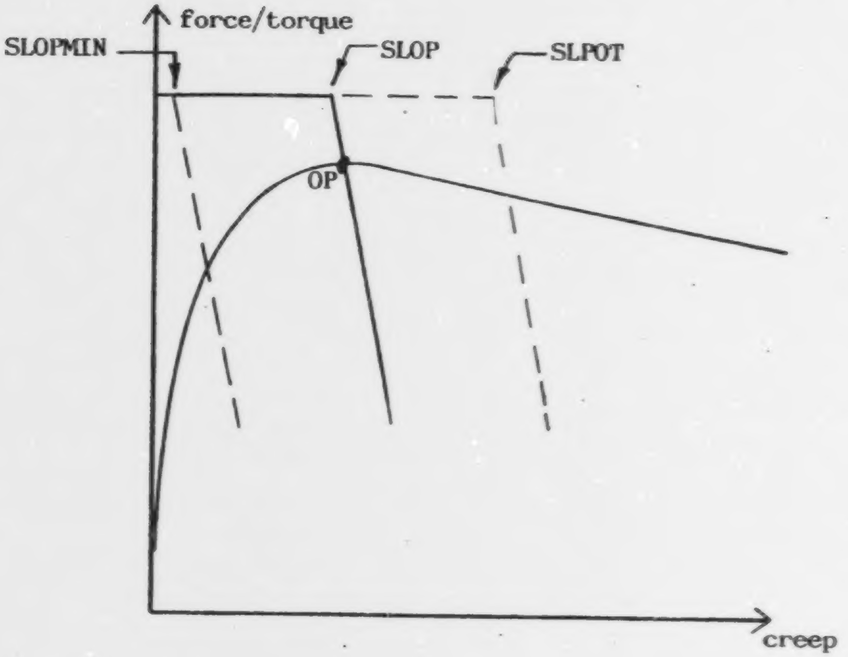


Fig. 5.5 Drive Characteristic with Maximum Microslip Control

5.4 Search strategy for optimal utilization of the friction

The task of the optimal friction utilization control is to set the maximum microslip SLOP such, that the operating point OP is in the maximum of the traction torque characteristic. It was already specified, that the search may stop close to the maximum, but at a positive slope, to provide optimal traction at the smallest possible microslip. It has to be kept in mind, that the characteristic is a band of values and that the maximum can shift any time to larger and smaller values.

To avoid too high slip values and interference with the operation at 'no slip' it is proposed to establish a range of slip to which the search is limited. That means illustrated with fig. 5.5, that the corner of the characteristic at SLOP can shift only up to the maximum value SLPOT and down to the minimum SLOPMIN. The lower limit is necessary to overcome the problems of limited accuracy of the locomotive speed measurement. From fig. 5.4 it can easily be seen, that an operation in the steep part of the traction force characteristic would require to set the value of SLOP within 0.1 percent or less to control the operation with the natural motor characteristic.
The maximum value SLPOT is set to limit the slip to the microslip range. The minimum value SLOPMIN is set at values of 1.5 to 2.5 precent and is effective, if the slip is lower then the slip SMIN. For slip values smaller than SMIN the search is not in effect and SLOP is equal to SLOPMIN. The locomotive works in this status like a slip limited system and the slip can not exceed SLOPMIN +

DSLM. If SLPOT is set to SLOPMIN, the maximum search is always disabled.

If SLPOT is larger than SLOPMIN and SMIN is exceeded, the search algorithm should start. At this moment the system operates at the limited microslip in the operating point OP in fig. 5.5 and SLOP = SLOPMIN.

The search is done in steps. The major activities of a step are shown in fig. 5.6. At the beginning the motor torque TMOT at the operating point is taken

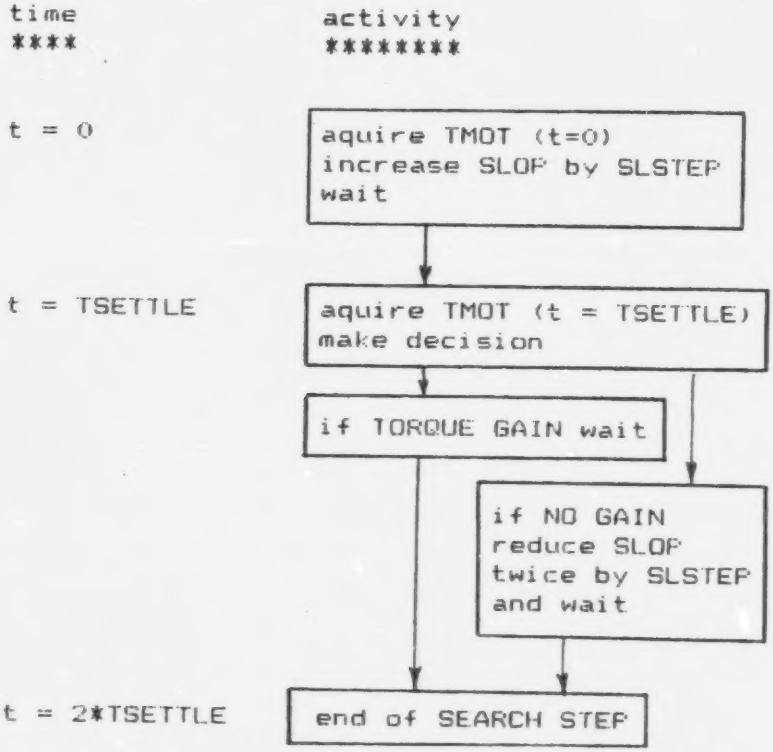

time
****
activity
****
t = 0
aquire TMOT (t=0)
increase SLOP by SLSTEP
wait
t = TSETTLE
aquire TMOT (t = TSETTLE)
make decision
if TORQUE GAIN wait
if NO GAIN
reduce SLOP
twice by SLSTEP
and wait
t = 2*TSETTLE
end of SEARCH STEP

Fig. 5.6 Search Steps

and the microslip limit SLOP is increased by a step SLSTEP. After the time TSETTLE, when a new operating point is reached the motor torque is taken again. The two motor torques are then compared and a decision is made, whether the increase in microslip limit SLOP resulted in an increase of traction or not.

If a gain in traction is sensed a new search step to further increase SLOP will start after the the time TSETTLE.

If there is no gain, or no sufficient gain, the microslip limit is reduced below its value at the start of the search by SLSTEP. After the system has settled at the new operating point with the reduced SLOP, a new search step will start.

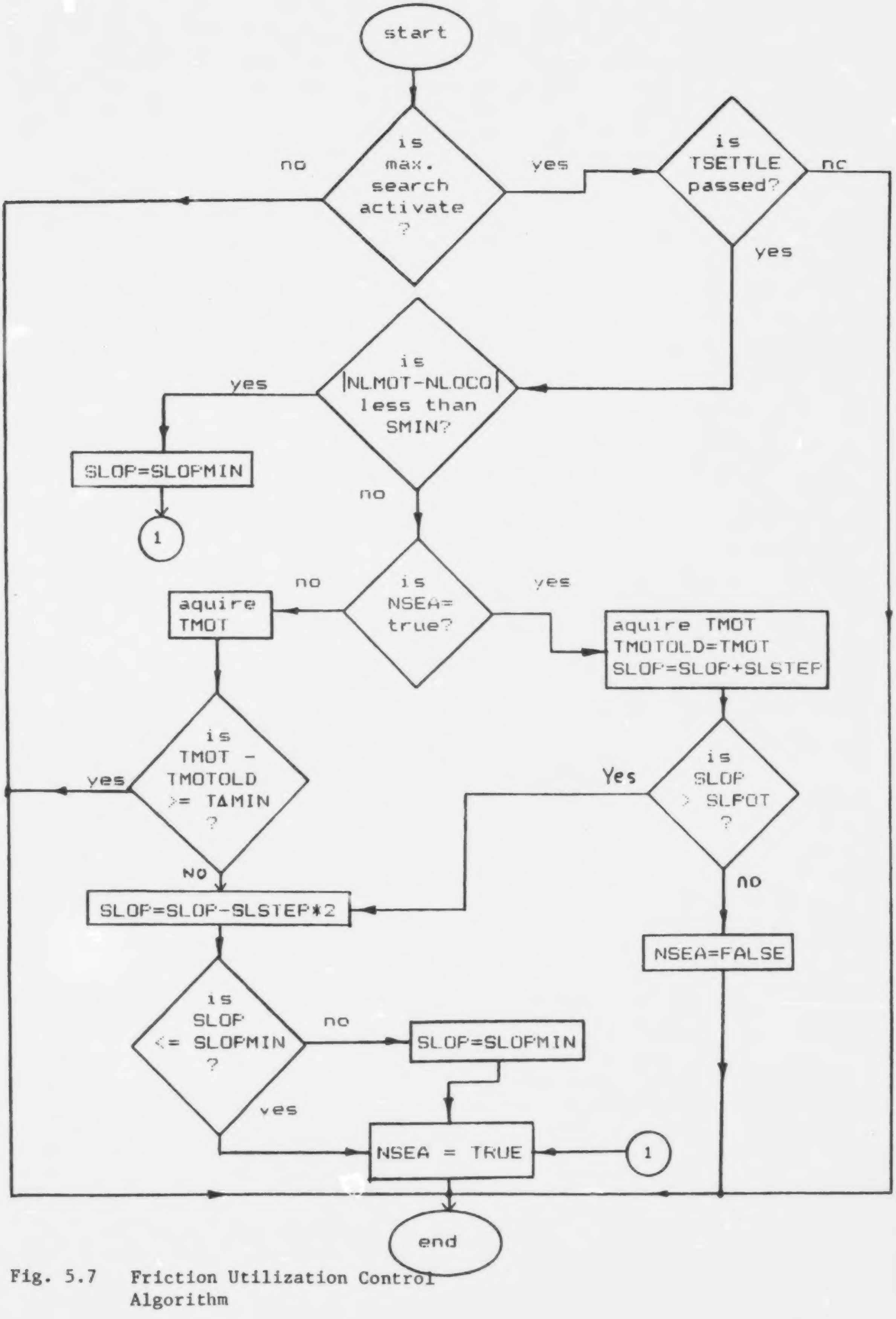

start
is max. search activate ?
no
yes
is TSETTLE passed?
nc
yes
is |NLMOT-NLOCO| less than SMIN?
yes
SLOP=SLOPMIN
1
no
is NSEA= true?
no
yes
aquire TMOT
aquire TMOT
TMOTOLD=TMOT
SLOP=SLOP+SLSTEP
is TMOT - TMOTOLD >= TΔMIN ?
yes
Yes
is SLOP > SLPOT ?
NO
no
SLOP=SLOP-SLSTEP*2
NSEA=FALSE
is SLOP <= SLOPMIN ?
no
SLOP=SLOPMIN
yes
NSEA = TRUE
1
end

Fig. 5.7 Friction Utilization Control Algorithm

The electric motor characteristic, as modelled for the locomotive, has a regulation of three percent for the per unit torque. This value and the value of two percent slip per unit traction force from characteristic #2 are higher than in reality. In reality both values are closer to one percent. This choice was made to observe quantitative differences between events despite the limited resolution of the display interface of the RIST system of about one percent, as described earlier.

With respect to the traction characteristic and the motor characteristic the microslip control parameters were selected. The minimum slip SMIN is 2.5 percent. Below this value the maximum search algorithm is not active and the slip limit is fixed at SLOPMIN. The value of SLOPMIN is chosen to be 2.8 percent. The maximum value SLPOT can be varied continuously during the simulation at a potentiometer between SLOPMIN and 30 percent.
The increment of the microslip SLSTEP during the maximum search is set to 0.4 percent. The selection results from the regulation of the motor and the tolerable maximum transient torque during a search step. The selected value results in a theoretical maximum transient torque in a search step of 13 percent.
The search time TSETTLE has to be selected such, that the drive can reach the new operating point resulting from the slip increment. On the other hand the search speed should be fast enough to go back to a lower slip value in a short time, to avoid operation at high slip, if it is not advantageous. The time was selected to be 0.25 seconds, which is a quarter of the time constant of the drive.
The difference in motor torque, which is considered to be enough gain in traction force to proceed with the microslip increase, was set to 0.15 percent. This depends on the selection of the microslip increment SLSTEP and the desired positive slope where the search should stop close to the maximum.

5.5.2 Simulation results

The simulation results are presented in the form of pictures taken from the display of the RIST system. The pictures show the observed variables in per unit values with 0.2 per unit per division in the vertical axis. The zero line is always one division from the bottom. The horizontal axis is the time. The variables shown are always listed from the bottom to the top at the left edge at time t = 0 sec.
The slip control of the MICAS controller has been eliminated, by lowering the sensitivity of the slip detection. To supervise the non interference of the MICAS slip control the traction power command W supplied by the MICAS system is displayed on most of the figures.
The following figures show different operations of the locomotive without microslip control, with microslip control and with microslip control with maximum traction force search to demonstrate the function of the proposed microslip control.

Figure 5.10 shows the locomotive speed NLOCO, the motor speed NMOT, the motor torque TMOT, the diesel engine speed ND and the power command W. NLOCO is always zero. ND and W are shown for control purposes only. The time is 2 sec per division.

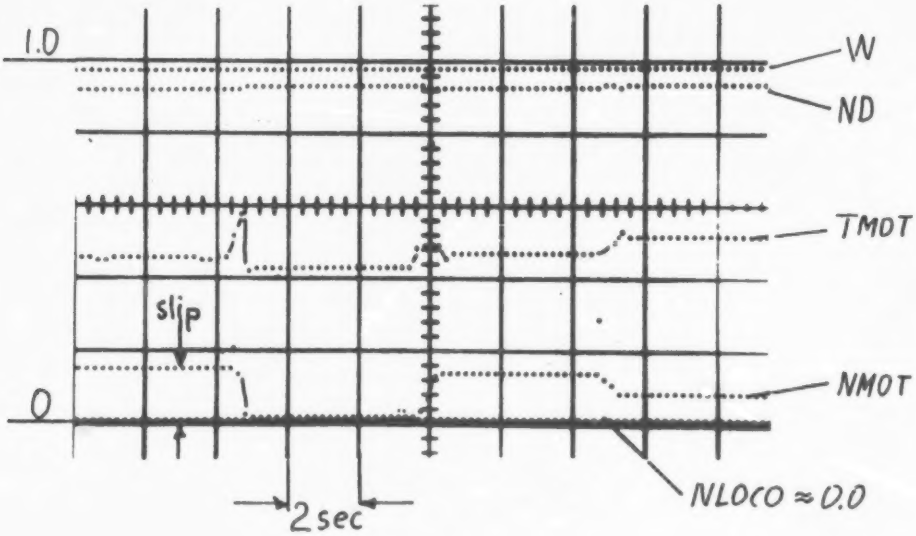


Fig. 5.10 Microslip Simulation Example

The locomotive operates in notch 8 with a heavy train (90 sec) on a steep grade. The traction force characteristic is #2 with TRGTO = 0.6. The result is that the available torque exceeds the friction by far and the locomotive will slip. The microslip control without maximum search is active. The control is set for always about 5 sec to four different values SLPOT. They are about 16%, SLOPMIN, about 15% and about 9%.

Figure 5.10 shows that the slip is limited to the set values. The slip is equal to NMOT, as NLOCO is zero. The motor torque operates in steady state at the values determined by the traction characteristic and TRGTO. According to this the highest torque occurs at the medium slip of about 9%. If the microslip limit is altered fast transients occur and the system stablizes at the new operating point in about 0.5 sec.

This simulation confirms the excellent slip limiting properties of the proposed microslip control.

Figure 5.11 shows the locomotive speed NLOCO,the motor speed NMOT, the power command W, the motor torque TMOT and the diesel engine speed ND. ND and W are shown for control purposes only. The time is 0.25 sec per division.

The locomotive accelerates in notch 7 with a heavy train (90 sec). The traction force characteristic is #2 with TRGTO = 1.0. The locomotive accelerates the train very slowly. At t = 0.5 sec the friction condition suddenly deteriorates, as TRGTO is changed linearly within 1 sec to 0.5. The locomotive would slip, but the microslip control, set at SLOPMIN for the slip limit, reduces the traction torque nearly parallel to the slip deterioration and keeps the slip limited at the commanded value.

The traction force characteristic is #2 with TRGTO = 0.6. The result is, the available torque exceeds the friction by far and the locomotive will slip. The microslip control without maximum search is set to SLPOT = 15% and controls the maximum slip at the set value. At t = 6 sec. the maximum search is activated and immediately reduces the slip and increases according to the traction characteristic the torque until it stablizes at t = 14 sec. at the maximum friction point.
Figure 5.14 shows the locomotive speed NLOCO,the motor speed NMOT, the motor torque TMOT, the diesel engine speed ND and the power command W. ND and W are shown for control purposes only. The time is 4 sec per division.

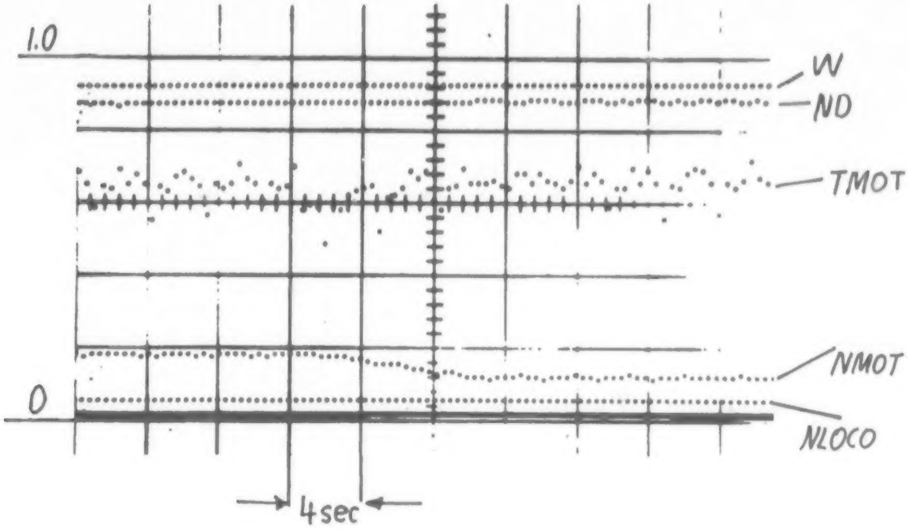


Fig. 5.14 Microslip Control - Backward Search

The locomotive operates in notch 8 with a heavy train (90 sec) on a steep grade. The traction force characteristic is #3 with TRGTO = 1.0. The result is, the available torque exceeds the friction and the locomotive will slip. The microslip control with maximum search is set to SLPOT = 15%. As the characteristic #3 has no maximum the slip is 15%, as limited by the microslip control.
To demonstrate the effect of sanding, which results usually in a traction force curve with a maximum at a low slip value, the traction force characteristic is switched from #3 to #2. This switching is initiated at the simulation control and is immediate. The change occurs at t = 12 sec. and it can be seen in fig. 5.14 that the microslip control searches immediately backwards and stablizes at the maximum friction with the lower slip.

The last three figures show the operation of the microslip control at the start up of the train.
The figures 5.15, 5.16 and 5.17 show the locomotive speed NLOCO,the motor speed NMOT, the power command W, which ramps up in 20 seconds to 100% , the motor torque TMOT and the diesel engine speed ND. The time is 4 sec per division.

The traction force characteristic is #2 with TRGTO = 0.9. The locomotive accelerates a heavy train on a grade (0.4). The throttle is shifted from notch 1 to notch 8. The diesel engine speed builds up and the traction power command W from the MICAS controller ramps up. The motor torque TMOT increases and finally exceeds the available friction and the locomotive enters into slipping.
Fig. 5.15 shows the case for the microslip control without maximum search. The microslip is limited to the set value SLPOT = 12% .

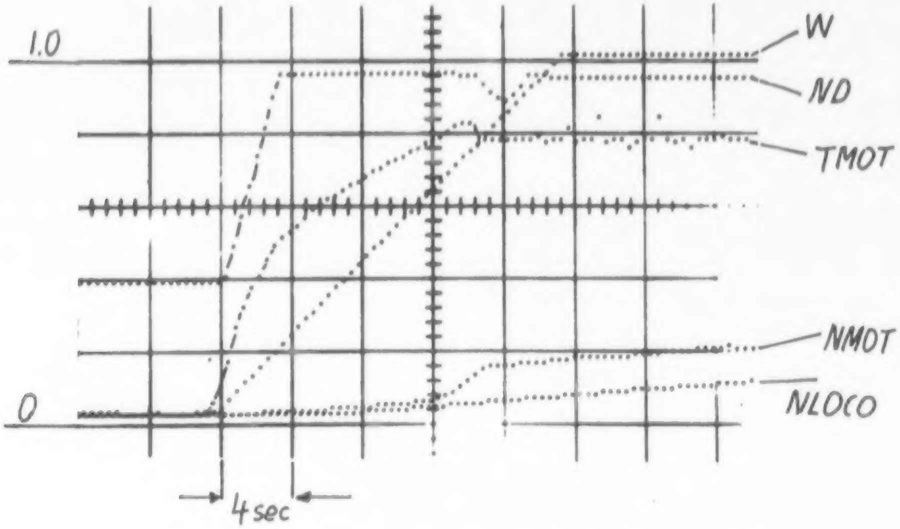


Fig. 5.15 Microslip Control without Maximum Search

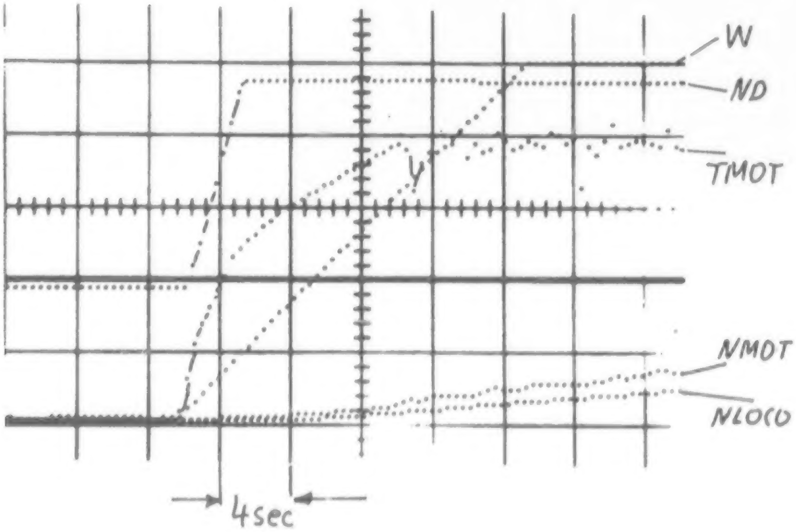


Fig. 5.16 Microslip Control with Maximum Search

Fig. 5.16 demonstrates the same case, but with the maximum search active. The system operates at the maximum friction, providing about the same torque at the lowest possible slip.
Fig. 5.17 finally shows a setting of SLPOT = SLOPMIN. This disables practically the maximum search and limits the slip to values smaller SMIN, but the torque is also lower and the available friction is not fully used.

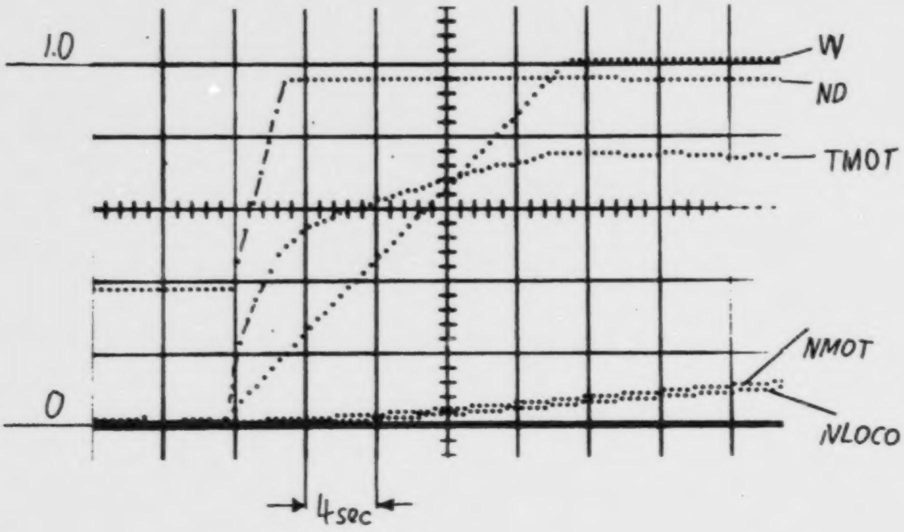


Fig. 5.17 Microslip Control without Maximum Search and with Limited Slip

The simulation results verify the proposed concept and demonstrate the good dynamic performance of the proposed microslip control. It demonstrates, that the requirements, specified at the beginning of the chapter, can be met with the microslip control and the proposed maximum friction search algorithm.

5.6 Implementation of the proposed microslip control

In section 5.5 the control concept was verified by simulation on the RIST system and it excellent performance was demonstrated. To implement the controls on the locomotive and the MICAS microprocessor control system it has to be studied how to get the feedback signals for the torque TMOT and the locomotive speed NLOCO, and how to connect and program the MICAS microprocessor control system.

5.6.1 Torque feedback

In the simulation the simulated motor torque was used as a feedback value in the maximum search algorithm to determine, whether the traction force increased or decreased changing the microslip. This signal is not available on the real locomotive. Furthermore it is not realistic to request the measurement of the average torque of the traction motors. Therefore another signal has to be found, which is easier to measure and can be used as the feedback signal to indicate a gain or loss in traction force.
The signal appropriate for this purpose could be the stator current of the traction motors. As the motors are fed from a bus, it is the current in the bus going to all motors.

The stator current of an induction machine is not always proportional to the motor torque. Only in certain areas and under certain conditions is it possible to consider the stator current proportional to the motor torque.
An induction motor fed with variable frequency and voltage by an inverter operates always under the condition that the rotor resistance is large compared with the rotor impedance. The rotor impedance is the product of the rotor frequency and the rotor leakage inductance. As this is the case, the rotor current and the magnetizing current are perpendicular to each other, and the motor torque is proportional to the product of rotor current and airgap flux.
The flux in the machine is controlled and known. In the base speed range the flux is often even kept constant. Under these circumstances the motor torque can be derived from the rotor current.
As the rotor current is not measurable, but the stator current is, the relation between stator current and rotor current would allow the use of stator current instead of the rotor current to determine the motor torque.
The stator current can be calculated as the square root of the squared rotor current plus the squared magnetizing current. Assuming that the magnetizing current is about 40% of the rated stator current, the stator current as a function of the rotor current for constant magnetizing current is shown in fig. 5.18. For rotor currents above 0.4 the relation between stator current and rotor current becomes quite linear and the torque is approximately proportional to the stator current.
From what was stated above it can be concluded that in the base speed range with constant flux operation the motor torque is nearly proportional to the stator current, if the torque is larger 40% of the rated torque. For operation with constant power this is not applicable, in this range it depends on the motor control used, whether it is possible to derive a torque proportional signal from the stator current and the operating point of the flux.

5.6.2 Implementation of the microslip control with stator current feedback

It should be recalled, that the microslip control using the natural characteristic of the motor does not need any torque feedback. This feedback is only used for the maximum friction operation with increased slip and automatic maximum search.
This increased slip condition will typically occur at high traction torque, for sure at motor torques above 40% . It is probably no severe limitation, if the automatic maximum search for increased microslip operation is limited to the base speed range. That means for operation at constant power the maximum search may be disabled. Concluding from this it is proposed to limit the automatic maximum search to the base speed range and to torque conditions, where the stator current is larger 50% . In this case the stator current can be used as a feedback signal for the automatic maximum friction condition search. An algorithm to detect the limit conditions, base speed operation and stator current larger 0.5 per unit, is easily to implement. The torque command control has the signal base speed or power operation already in its algorithm and it has only to be provided to the maximum search algorithm as a condition to activate the maximum search. The other condition is derived from the stator current feedback.

It should be stressed, that the above limitation affects only the automatic maximum search. It is still possible to set the microslip limit SLOP to a higher value to operate at increased microslip also in the constant power operation range.

5.6.3 Locomotive speed measurement

The locomotive speed feedback is crucial to the microslip control. A high accuracy is required and the dynamic of the feedback pass should not determine the dynamic quality of the microslip control. Studies at the simulation indicated that the feedback time constant should not exceed 0.1 seconds, for it be of minor influence on the dynamic of the microslip control.

Assuming an accuracy of 0.5% and a delay of 0.1 sec. it is possible to derive the requirements for the use of an incremental shaftencoder on a non driven axle for the speed feedback. Using an incremental shaft encoder with m marks at the circumference and a counter to count the pulse with a counting time tc the accuracy acc of the speed in percent is :

$$acc = 100/(m \times tc \times maximum\ speed)$$

From this it can be derived, that i.e. for a maximum axle speed of 800 rpm, an accuracy of 0.5% and a counting time of 0.1 sec. an encoder with 150 marks per circumference is required. This applies if only rising or falling pulse edges are used for counting. Depending on the accuracy of the equidistance of the encoder marks, processing methods other than straight counting can be used, which provide superior accuracy and time constant for the feedback.
If feedback systems based on radar are used, then accuracy and dynamic properties should be carefully examined.

5.6.4 Implementation of the control on the MICAS system

To implement the microslip control on the MICAS system instead of on the VME computer, it is necessary to feed to the MICAS interface the locomotive speed NLOCO. The MICAS controller provides then the slip limiting value NLOCO+SLOP. These interface values have to be transferred with high resolution, as increments of SLOP of 0.4% were used in the maximum search.
This interfacing is not possible with the MICAS microprocessor provided for the project, as the available interface, an analog input has only 8 Bit resolution. The MICAS system was connected as described, and it proved, that the resolution was not sufficient. To improve the situation the MICAS microprocessor system provides analog inputs and outputs with 12 Bit resolution, which should be sufficient, but were not available for the project.
Considering the high resolution it is not recommended to connect the MICAS microprocessor as described above. It is suggested that the microslip control be implemented in the same processor as the inverter control. This strategy avoids the double conversion of digital to analog and back and therefore the full resolution of the digital processing is maintained. Avoiding the double conversion is also desirable to reduce the possibility of noise interfering in these sensitive signals.

To prepare the implementation of the microslip control on the MICAS microprocessor system, and to see, whether the available MICAS software is sufficient to implement the maximum friction search algorithm, a MICAS program for the microslip control, particular for the maximum search was written. This program implements the logical structure shown in the flow chart in fig. 5.7. The MICAS program blocks and their connection is shown in fig. 5.18 and fig. 5.19. The program listing is attached in appendix A5.

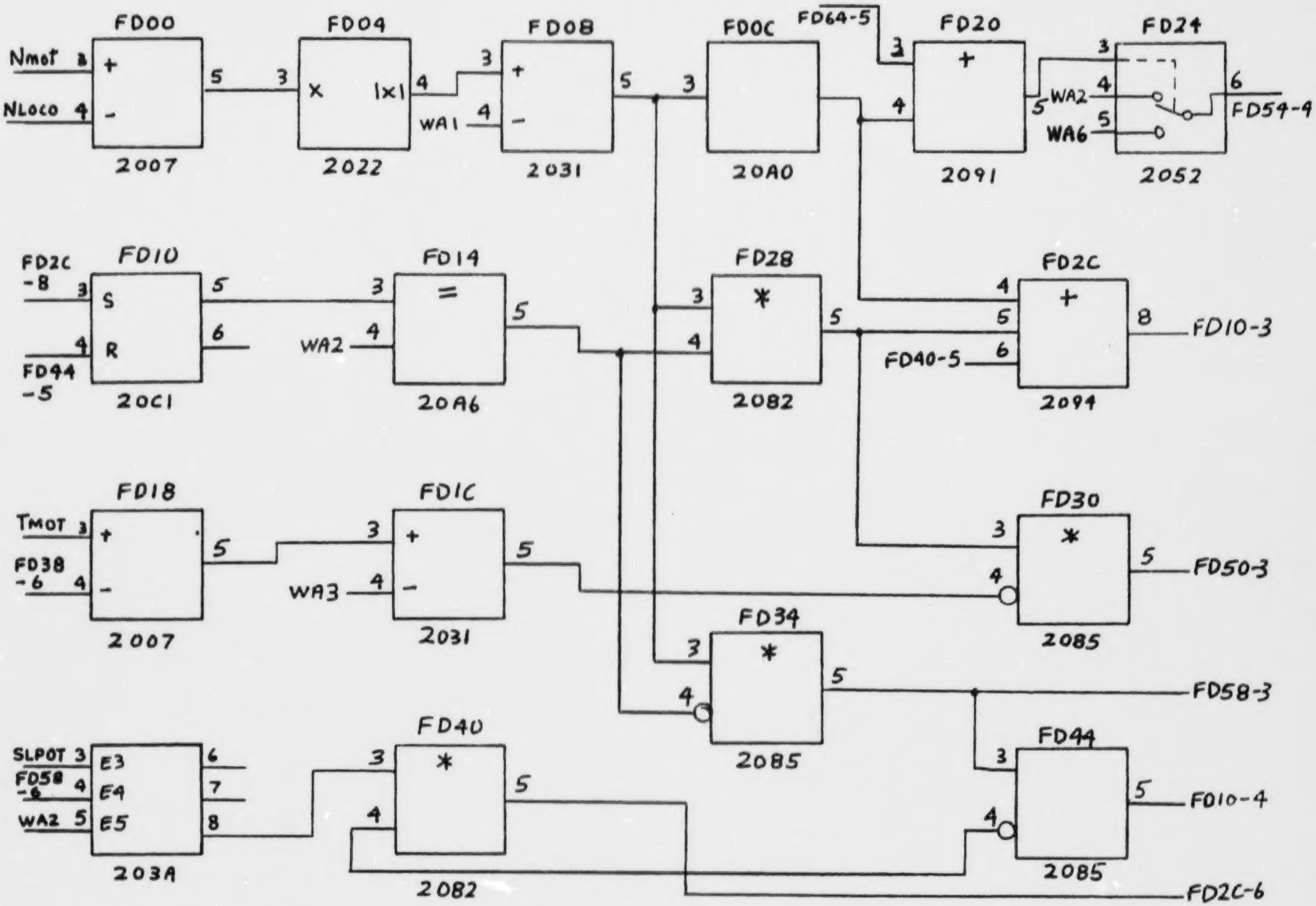

FD00
Nmot 3
NLoco 4
+
-
5
2007
FD04
3
X
|X|
4
2022
FD08
3
+
WA1 4
-
5
2031
FD0C
3
20A0
FD64-5
FD20
3
+
4
5
2091
FD24
3
WA2 4
WA6 5
6
FD54-4
2052
FD2C
-8
FD10
3
S
4
R
FD44
-5
5
6
20C1
FD14
3
=
WA2 4
5
20A6
FD28
3
*
4
5
2082
FD2C
4
+
5
FD40-5 6
8
FD10-3
2094
TMOT 3
FD18
+
FD38
-6 4
-
5
2007
FD1C
3
+
WA3 4
-
5
2031
FD30
3
*
4
5
FD50-3
2085
FD34
3
*
4
5
FD58-3
2085
SLPOT 3
E3
FD58
-6 4
E4
WA2 5
E5
6
7
8
203A
FD40
3
*
4
5
2082
FD44
3
4
5
FD10-4
2085
FD2C-6

Fig. 5.18 MICAS Program Blocks and Connections

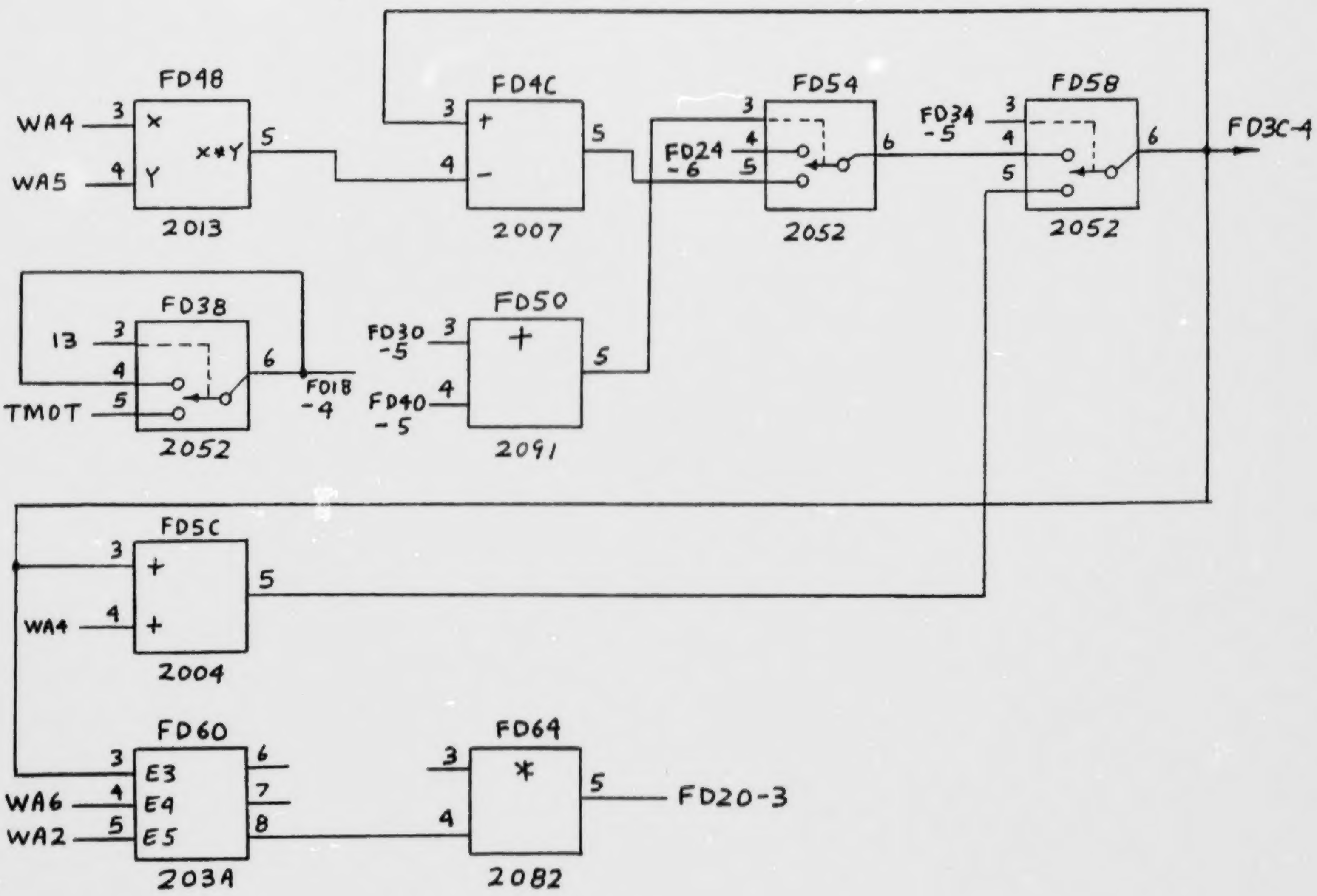

FD48
WA4
WA5
3
4
X
Y
X*Y
5
2013
FD4C
3
4
+
−
5
2007
FD54
3
4
5
FD24
-6
6
2052
FD58
FD34
-5
3
4
5
6
2052
FD3C-4
FD38
13
TMOT
3
4
5
6
FD18
-4
2052
FD50
FD30
-5
FD40
-5
3
4
+
5
2091
FD5C
3
4
WA4
+
+
5
2004
FD60
WA6
WA2
3
4
5
E3
E4
E5
6
7
8
203A
FD64
3
4
*
5
FD20-3
2082

Fig. 5.19 MICAS Program Blocks and Connections

5.7 The old MICAS slip limit control with rate of change sensing

The MICAS microprocessor control system provided for the project included some software for the initial locomotive control. This software contains a slip limit control, which detects the slipping by sensing the rate of change of the motor speed NMOT. Depending on the rate the system is supposed to lower the traction power command. To see how this slip control handles the test functions used for the verification of the microslip control, it was tried to use this control.

It must be stated that a comparison is not really possible, as the systems are quite different and a system with feedback of the locomotive speed is superior. Furthermore the provided system was not programmed with realistic values and the programming, which was done is not an optimization. In short, the MICAS slip control program as used is not state of the art.

Despite the above facts it is interesting to see, how this control reacts. The following pictures show the action of the control. It becomes obvious that a certain rate of change is needed, before the control starts to reduce the traction power command W.

If the slip is small, when this rate is sensed, the system is able to limit slipping. This is always the case, when the inertia of the train is small. For larger inertia the slip is large when the rate of change triggers the control, and the control oscillates and is not able to limit the slip.

Figure 5.20 shows the locomotive speed NLOCO, the motor speed NMOT, the traction power command W, the motor torque TMOT and the diesel engine speed ND. NLOCO and NMOT are nearly identical. The time is 4 sec per division.

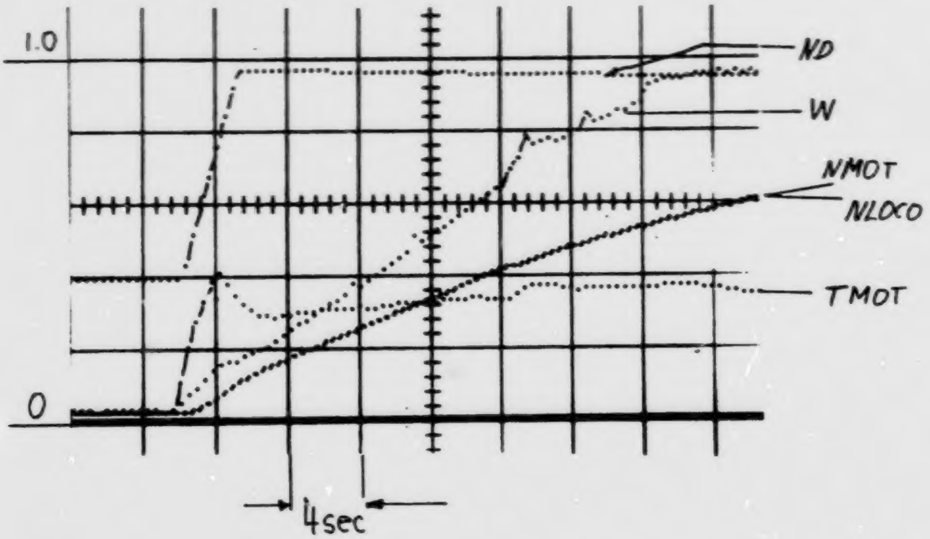


Fig. 5.20 Slip Control with Rate-of-Change Sensing

The throttle is shifted from notch one to notch eight to start up a light train (10 sec.). The traction force characteristic is #2 with TRGTO = 0.9. The motor and train speed rise fast and no slipping occurs. But the rate of change is too fast, so that two seconds after the begin the control reduces the rate of change of the power command W. From the observed event one would state, that the sensitivity of the rate of change sensing is set too low.

Figure 5.21 shows the locomotive speed NLOCO,the motor speed NMOT, the traction power command W and the diesel engine speed ND. The time is 4 sec per division.

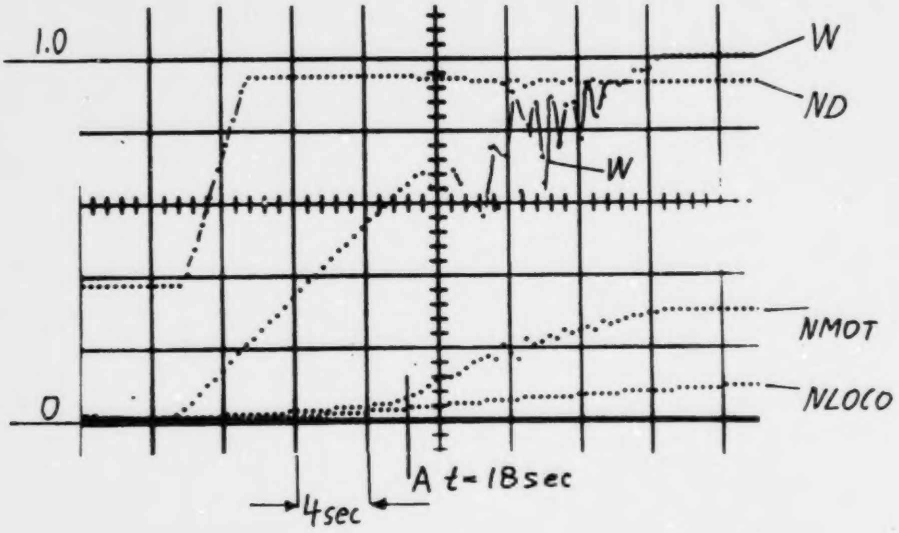


Fig. 5.21 Slip Control with Rate-of-Change Sensing

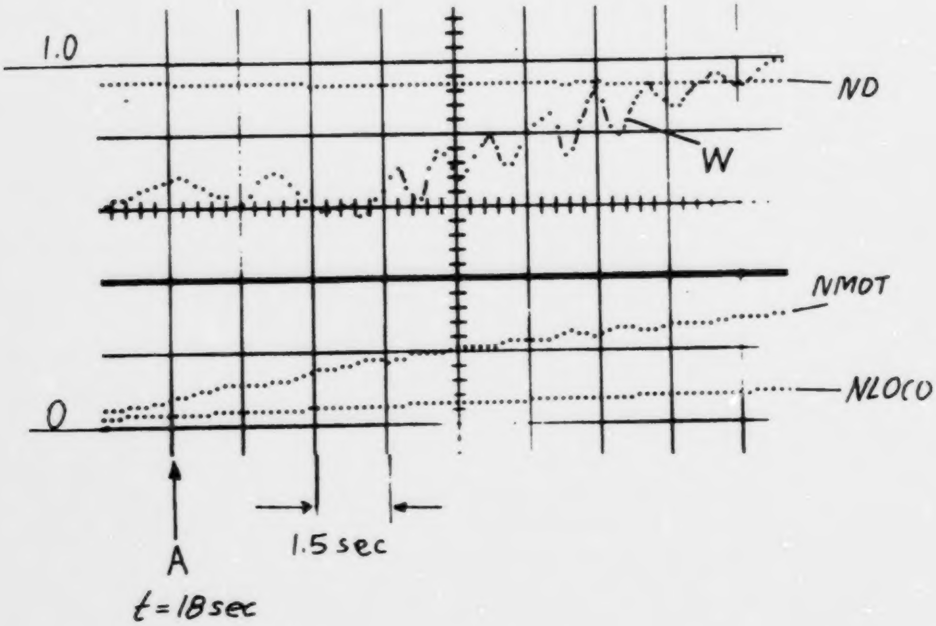


Fig. 5.22 Slip Control with Rate-of-Change Sensing

The event is the same as in the fig. 2.20 but the train is heavy (90 sec.) and is started at a grade (0.4). The train speed builds up slowly until the limit of the friction is reached at t about 18 sec.. The locomotive starts slipping. At about t = 20 sec. the slip is already 6% and the rate of change is now that high, that the slip control lowers the power command W. But the slip control is not able to stablize the slipping. Figure 5.22 shows an expended view of fig. 5.21 from t = 18 sec. to t = 22 sec. with 1.5 sec. per division.

Judging from this event the sensitivity of the rate of change sensing is set to high, as it senses the slipping too late. This statement contradicts the earlier observation in fig. 5.20 and indicates the difficulties this type of control has to find a proper adjustment of the sensitivity levels.

Figure 5.23 shows the locomotive speed NLOCO,the motor speed NMOT,

the traction power command W, the motor torque TMOT and the diesel engine speed ND. The time is 1 sec per division.
The locomotive is operating in notch 7 with the traction force characteristic #2 and TRGTO = 1.0. The train inertia is low (10 sec.) and a braking force is applied to the train, which keeps the speed at a low value.
The transient is started by releasing the brakes and reducing about 0.5 sec. later the available friction by reducing TRGTO to 0.5 within a second. This makes suddenly a high acceleration torque available, which as soon as the friction is reduced will start a slip event.
The MICAS slip control is able to handle this event, as the rate of change is high and the slip does not build up.

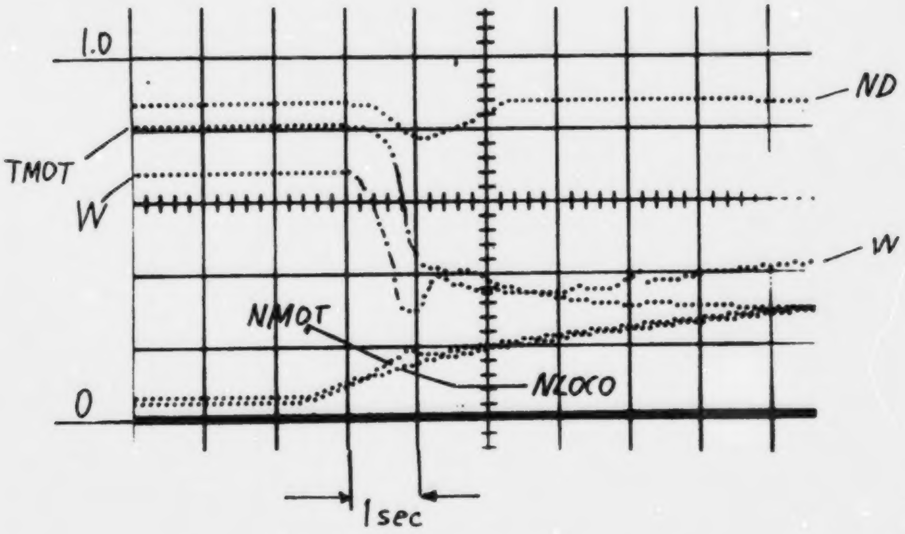


Fig. 5.23 Slip Control with Rate-of-Change Sensing

Figure 5.24 shows the same event, but the train inertia is increased (50 sec.) and the control fails completely. This indicates once more, that this type of

control has considerable difficulty managing variable train inertia.
In fig. 5.24 the motor torque TMOT is not shown, to avoid confusing it with the power command W.

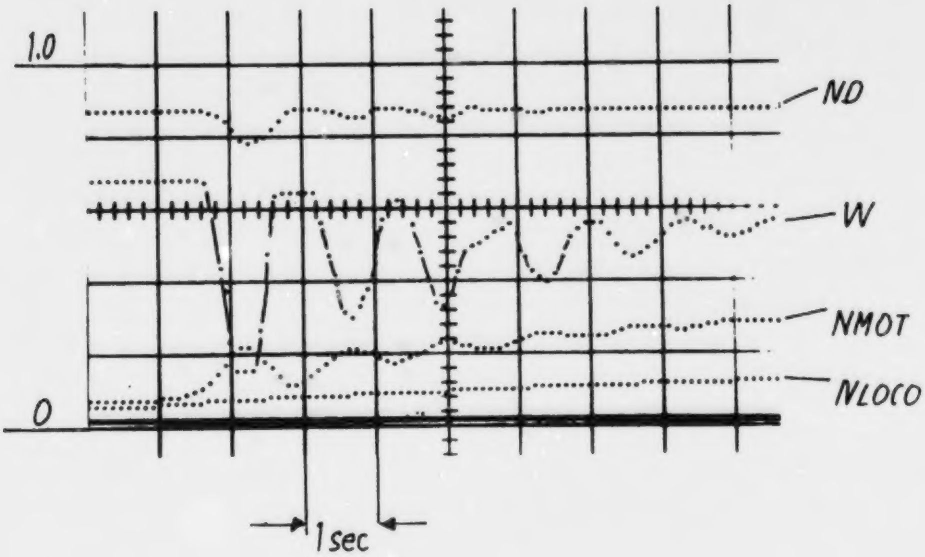


Fig. 5.24 Slip Control with Rate-of-Change Sensing

As stated earlier a direct comparison with the microslip control is not fair, but to demonstrate the vast difference in performance, the same events as shown in fig. 5.23 and 5.24 are repeated with the proposed microslip control. The microslip limit SLOP is set to 3% .
Figure 5.25 shows the microslip control response comparable to fig. 5.23. Fig. 5.26 is comparable to fig. 5.24 but the train inertia was even more increased (90 sec.). This shows, that the microslip control has no problems accommodating a wide range of train inertia.

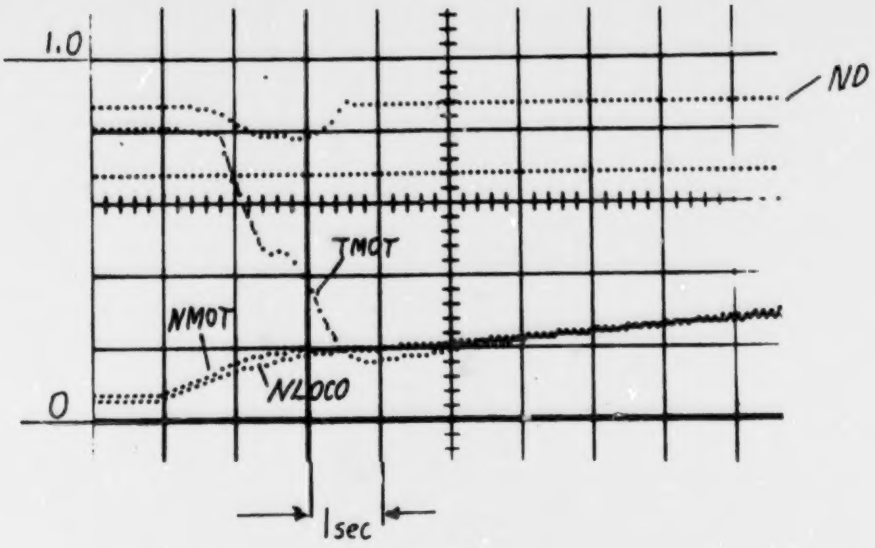


Fig. 5.25 Slip Control with Rate-of-Change Sensing

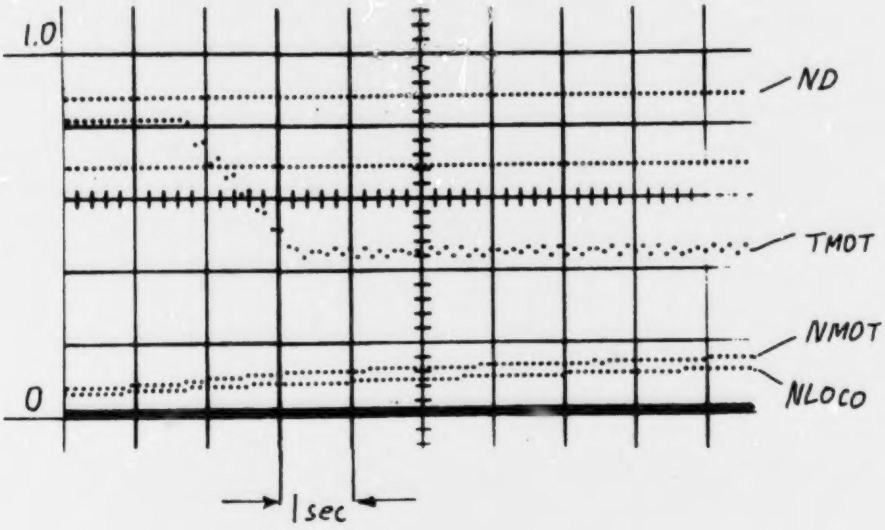


Fig. 5.26 Slip Control with Rate-of-Change Sensing

In fig. 5.23 to .25 the diesel engine speed is affected by the transient. This is the result of a fast motor speed increase, which loads the diesel engine faster, as the turbo charged engine can rise the output torque. The result is a transient overload to the diesel engine. This problem, and a proposal of how to control it, is discussed in chapter six.

5.8 Comparison to other potential slip controls with different drives

In proposing a new microslip control for a diesel electric locomotive with ac drive, the potential of this control should be compared to other existing or potential systems. It is not possible to compare such controls in detail, as internal knowledge of the controls would be required. This internal data are normally treated as company or manufacturer secret and are not available. But it is possible to judge the potential of the different electric drives to provide a high performance slip control.
Whether this slip control is then designed as a slip limit control or as a microslip control and whether a locomotive speed feedback is used defines only control categories, which are applicable to each drive.

5.8.1 General influence of the drive characteristic

Of major influence on the utilization of the available friction is the traction force characteristic and in particular the spread of the characteristics between the axles of the locomotive.
The influence of the electric drive characteristic is determined by the type of drive control and motor used. The drive control can be a single motor control, or a common feeding of all motors from a common bus. The drive control is normally based on power electronic controllers and therefore the dynamic performance of all drive controls is sufficient for the diesel electric traction.

The principle situation is shown in fig. 5.27. The figure shows the situation

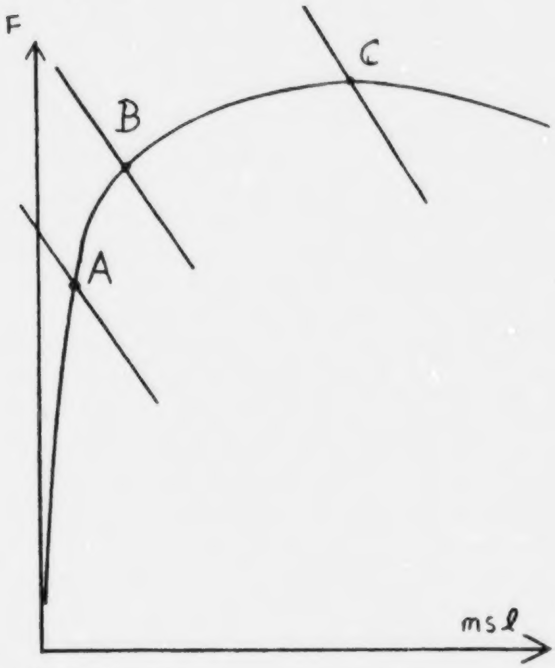


Fig. 5.27 Drive Characteristic - Single Axle

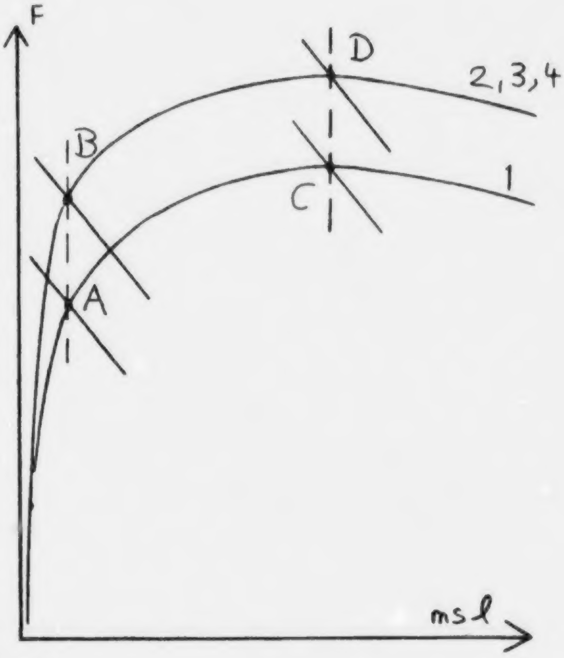


Fig. 5.28 Drive Characteristic - Multiple Axles

for a single axle or for the case where all axles have identical traction force characteristics. The intersection of a traction force characteristic and a drive characteristic determine the operating point A, B or C. The setting of the drive control determines at which point the system operates.

In reality the traction forces between the axles differ and a situation as shown in fig. 5.28 may occur. The axles 2,3,4 can provide more traction than the axle 1. In this case the drive control becomes important. It can be different between single motor or axle control and common feeding of all motors from one bus, which is controlled.

The situation with single axle control is shown in fig. 5.29. To use the available traction optimally without increased slip, the axles 2,3,4 would be controlled at the operating point B. The axle 1 would be controlled at the operating point A. For maximum friction use with increased slip the equivalent operating points would be D for the axles 2,3,4 and C for axle 1. That means the single drive is controlled such that it operates at the same speed or slip as all other drives, but with different torque.

The usual situation in today's locomotive is feeding of all motors from a common bus, which is controlled. This situation is shown in fig. 5.29 and 5.30.

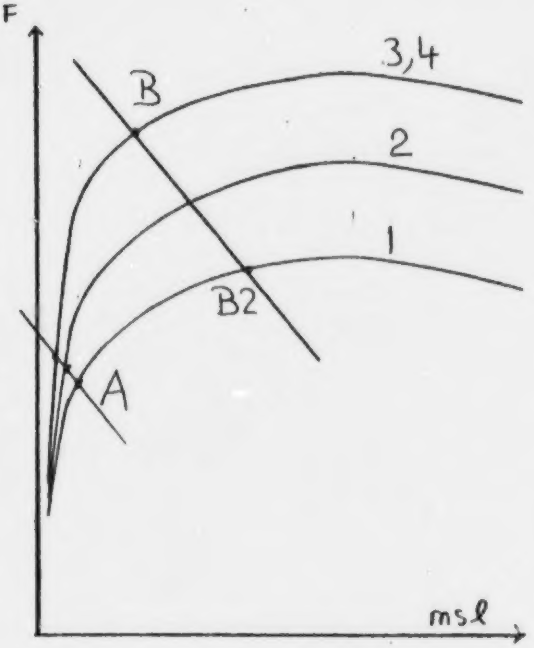


Fig. 5.29 Drive Characteristic - Common Bus Control

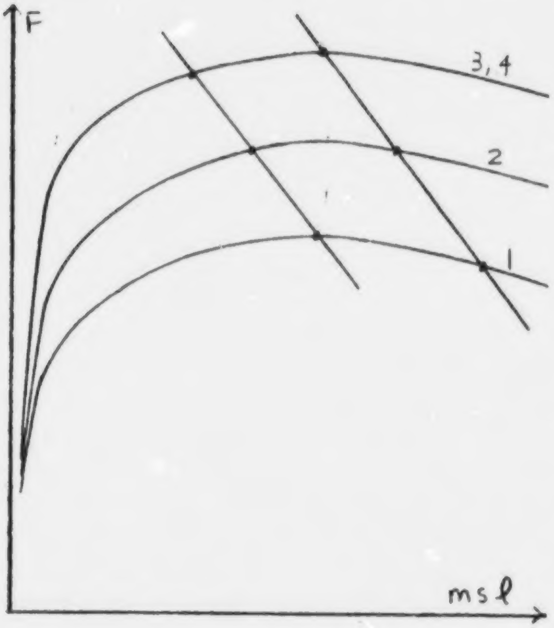


Fig. 5.30 Drive Characteristic - Common Bus Control

There is only one drive characteristic. If this is controlled to operate in the point A in fig. 5. to operate in the point A in fig. 5.29, then the available traction of the axles 2,3 and 4 is not fully utilized. If on the other hand the control is set such that the operation point is B in fig. 5.29, then the axle 1 runs in point B2 at a high slip and this may be already the case of single axle slipping.

Figure 5.30 shows a similar case for increased microslip and it is obvious that the maximum friction cannot be utilized at all axles at the same time. Furthermore the difference in slip is considerable between the different axles.

In the case of common bus feeding the slope of the drive characteristic becomes important. This slope is determined by the motor characteristic. Fig. 5.31 repeats the situation from figs. 5.29 and 5.30, but this time assuming

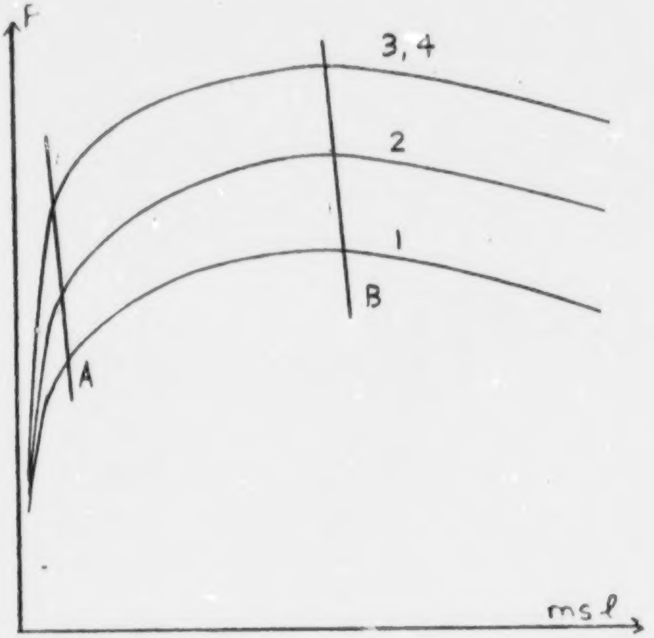


Fig. 5.31 Effect of Steep Motor Characteristic

a motor characteristic, which is very steep. It becomes obvious that with such a characteristic the available traction on each axle can be fully utilized and the slip difference between the different axles is so small, that one is tempted to state that they operate at the same speed. This situation comes very close to the situation in fig. 5.28 for single axle control. It should immediately be stressed that this situation occurs only, if the slope of the motor characteristic is as steep as the traction force characteristic in the non slipping section. This means a regulation of the motor characteristic of about one percent microslip per rated motor torque.

The result of the above considerations is that if the traction available at each axle differs, then a full utilization of the traction at all axles at the same time is only possible with single axle control or for common bus feeding with an extremely steep motor characteristic.

5.8.2 Friction utilization potential of different drives

There are mainly three electric drives used in rail transportation. The dc series motor with voltage control and sometimes additional field control, the dc motor with separate excitation with armature voltage and separate field control and the ac induction motor with squirrel cage rotor and stator frequency and stator voltage control

The motor characteristics of these drives are shown in fig. 5.32. The series motor #1 has a soft characteristic, in which slope depends on the operating point. The separately excited dc motor #2 has a constant slope, which is steepest at full field. The induction motor has the best regulation. For the size of traction motor its regulation is several times better than the one of the separately excited dc motor.

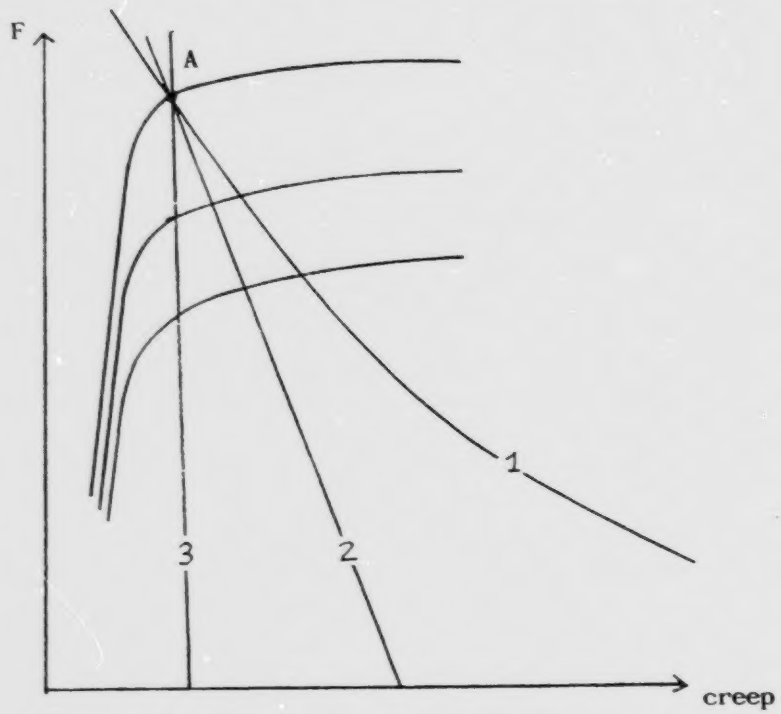


Fig. 5.32 DC and AC Drive Characteristics

Studying the regulation of the different motors quantitatively, it becomes obvious that only the induction motor has a regulation, which is close to the steepness of the traction characteristic in the non slipping part. Only this regulation of about one percent creep per full rated torque allows to utilize the maximum traction at each axle at the same time with common bus feeding for the traction motors. DC drives with common bus control cannot achieve the same utilization due to their softer characteristic.

To achieve the same utilization with the dc drive a single axle control is required. This is possible by individual field control at the separately excited dc motor and to a certain degree for the series motor. The difficulty with the series motor in single axle control is that the field currents are large and its general control characteristic is operating point dependent.

By comparing locomotives with dc drives having single axle control to locomotives with ac drives and common bus control, it should not be overlooked that the ac drive has the option of single axle control as well.

6. Diesel Engine Overload Protection and Maximum Power Utilization

The studies of the diesel locomotive system showed clearly the possibility of overloading the diesel engine. This is particularly true for a charged engine. This overload occurs mainly during transients. It can also be caused by a malfunction of the diesel or parts of the drive control.

Most overload conditions during transients can be avoided, by limiting the rate of change of the transients. These is possible for transients caused by commands; it is more difficult for transients caused by disturbances like slipping.
Furthermore, for the microslip control it is desirable to utilize the full dynamic performance of the electric drive for the small signal behavior to stabilize the system at the operating point. Large signal changes, which can cause diesel overloading, should be slowed down.

To protect the diesel engine against overload, a control is proposed which monitors the diesel engine and if overload is detected reduces the loading. The load reducting is done by intercepting and reducing the power command to the electric drive. This reduction can be temporarily or continuous.

6.1 Concept of the overload control

The principle idea of the control is to monitor the diesel speed control and, if the control can not maintain the diesel speed, to assume overloading and to reduce the traction power command.

In normal operation the diesel speed regulator or governor maintains the diesel speed error NERR smaller than a maximum error NERRMAX. The diesel speed error is the difference between the diesel speed command and the feedback value of the diesel speed. If the load varies and the error increases the torque to maintain the diesel speed. The torque increase is the result of supplying more fuel to the diesel. In the case of overload, the regulator is not able to maintain the diesel speed, as the fuel supply is already at its maximum. This situation means the closed loop of the speed control is open, as the limit of the fuel supply is reached.

The criterion that fuel supply requested by the governor exceeds fuel supply limit indicates the speed loop is open. This may be a short transient situation and the diesel speed error is kept in it's limits, due to the diesel engine and generator inertia. But if the diesel speed exceeds the maximum error, then a load reduction is indicated. This load reduction will depend on the value of the error and its duration. It also can be considered to take the rate of change of the error into account.

6.2 Simulation of an overload protection control

The principle structure of the control is shown in fig. 6.1. The dashed part is from the diesel model in fig. 3.7.

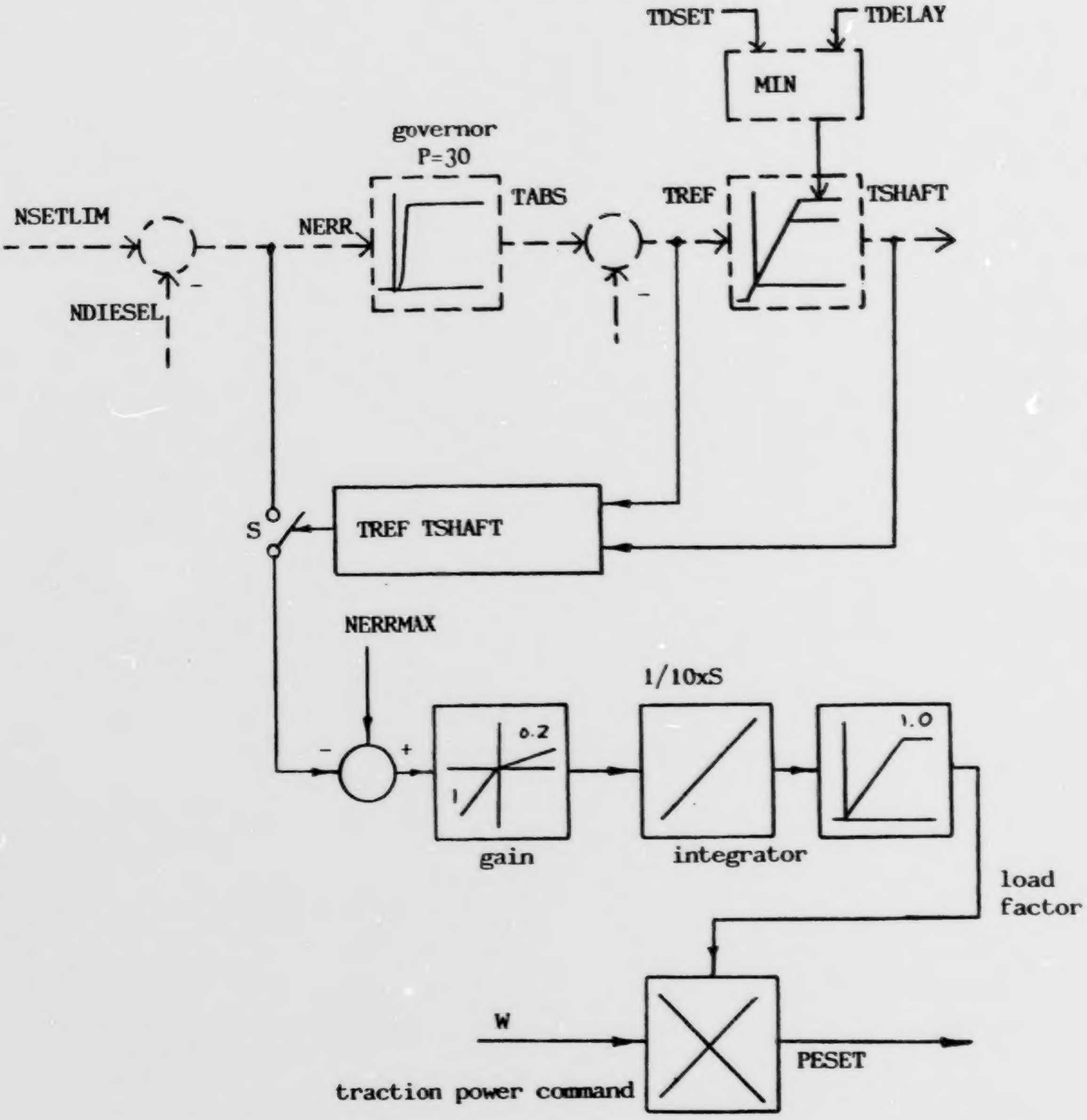

TDSET
TDELAY
MIN
governor
P=30
TABS
TREF
TSHAFT
NSETLIM
NERR
NDIESEL
-
-
S
TREF TSHAFT
NERRMAX
1/10xS
-
+
0.2
1
1.0
gain
integrator
load factor
W
traction power command
PESET

Fig. 6.1 Overload Protection Control Simulation

The control monitors the the condition TREF larger TSHAFT. If the request from the governor TREF exceeds the actual diesel shaft torque, the limit is reached and the speed loop is open. In this case the switch S is closed and the speed error NERR is compared to the maximum allowed error NERRMAX.
If NERR is smaller than NERRMAX, the value of load factor is always one and the traction power command is not reduced. If NERR is larger than NERRMAX, the value of load factor is reduced. The amount of reduction is dependent on the size of the error, the gain and the time constant of the integrator. The gain may be different for the reduction and reapplication of the traction power command.

A result of the simulation is shown in fig. 6.2. The figure shows from bottom to top the diesel shaft torque TSHAFT, the diesel load torque TDLOAD, the diesel engine speed ND, the power command W and the value of the load factor. The time is 1 sec. per division.

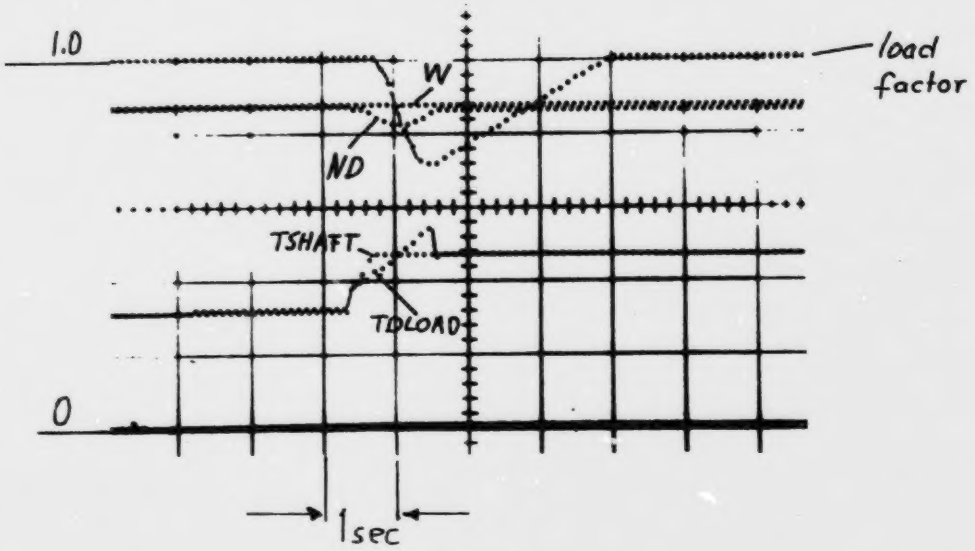


Fig. 6.2 Overload Protection Control Example

At t =3.3 sec. the load torque TDLOAD is ramped up in 0.3 sec. from 0.32 per unit to 0.48 per unit. The shaft torque can not follow that fast due to the dynamic of the turbo charger. As a result the diesel is overloaded and the diesel engine speed drops. As soon as the speed error exceeds 3% the load factor is reduced. In this case the load factor has no influence on the load torque and therefore the value of the load factor is reduced as long as the speed error is larger than NERRMAX at t = 4.5 sec.. Then the value of the load factor increases again to one.

In practice the monitoring of the speed control loop will not be done by monitoring the difference of the requested torque TREF and the diesel shaft torque TSHAFT, as TSHAFT is difficult to acquire. Instead the difference of TREF and the maximum diesel torque command TDSET may be used. TREF is proportional to the requested fuel and TDSET is proportional to the maximum possible or allowed fuel. These values should be easily acquired.

That the values TREF and TDSET can be used is demonstrated by the simulation shown in fig. 6.3. It shows the diesel shaft torque TSHAFT, the torque requested by the governor TREF, the maximum diesel torque TDSET and the diesel engine speed ND. The values TSHAFT and TREF are identical in steady state at the begin and the end of the transient. The time is 1 sec. per division.
The diesel engine is operating at notch 7 with a load of 0.35 per unit. At t = 2 sec. the load torque is suddenly changed to 0.45 per unit. The shaft torque can not follow that fast and ramps up limited by the rate of change the charger permits. As a result the diesel is momentarily overloaded and the diesel engine speed starts to drop. The governor tries to correct this by requesting more torque and TREF shoots up steeply to one per unit, exceeding the maximum diesel torque allowed TDSET = 0.75.

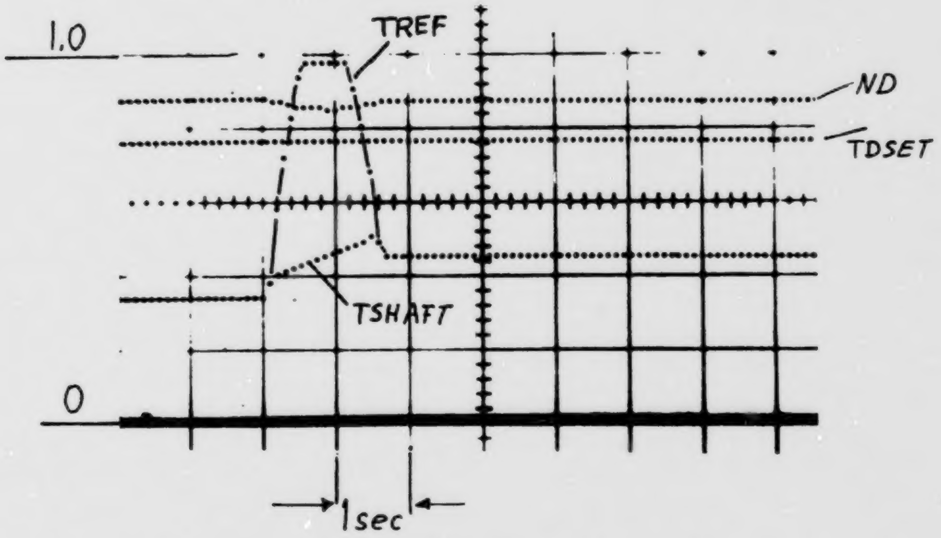


Fig. 6.3 Overload Protection Control Example

The condition TREF larger TDSET prevails nearly as long as the condition TREF larger TSHAFT and can therefore be used as a signal indicating the open loop condition of the speed loop.

6.3 Maximum diesel power utilization with the load factor control

Under certain conditions it may be desirable to use all available diesel power to get the maximum possible traction power from the locomotive. This is normally difficult, as the maximum diesel power varies with engine conditions, weather and other conditions. Even if the maximum diesel power is known, the

auxiliary power taken from the diesel is not known and the efficiency of the generator and the electric drive vary too.
Instead of difficult and unreliable calculations to determine the maximum available traction power due to the limited diesel power it is suggested that the diesel be loaded until the overload limit is reached and to stabilize the point by a load factor control as suggested above.

7. Monitoring and diagnostics

The study should investigate how to use the MICAS system for data acquisition. Data acquisition is used as general term for the following tasks:

acquisition of the data
storage of the acquired data
processing of the acquired data
storage of the processed data for documentation
providing documents or displays of the processed data

7.1 Tasks and capabilities of the MICAS system

The MICAS system is the microprocessor based control system including the software, as it is available to the University of Toronto. The hardware consists of a central processing unit based on the 8086, memory based on EPROM and RAM in low density with 10-20K byte per memory card, a serial port for communication and different I/O cards for the input and output of analog and digital values.

The software includes the operating system, the control-blocks with user specified parameters and a small monitor. The operating system executes the control-blocks periodically or event triggered. The monitor together with the serial port allows to do simple tests and to load data into the memory.

Checking which of the data acquisition tasks 1-5 can be done or partially done by the MICAS system has the following result:

Acquisition of data

All input values, all intermediate process values and certain output values are accessible as a digital word in a memory location. Some output values are not available in memory, but can be made available using existing MICAS software. Therefore all variables in the system are accessible for acquisition.

Storage of acquired data

The memory capacity of the MICAS system is limited and the memory fills up quickly, if continuous data storage is used. The available memory is RAM memory, that means the content is lost with power down. Due to the limited memory capabilities there is no software for convenient use for continuous data storage.

Processing of acquired data

The only way to process data is to use the available control-blocks. But this is not sufficient and very laborous. There are no easy ways to sort, to reduce data and to get values like the average or the mean value of a number of values. If this has to be done, new blocks must be developed for each task and everything is done in assembly language.

There is no possibility for the user to program different processing programs for varying needs in a familiar high level language. The real time operating system of MICAS is not made to handle compilers or general computing as one is used to on mainframes or small general computers.

Storage for documentation

The typical storage medium for this task is a disk or tape. The information is not lost on power off and the capacity is in hundreds of kilo bytes or in mega bytes. MICAS does not have these facilities. If MICAS should be extended to handle this task, not only considerable hardware has to be added but a new operating system is required and appropriate user software. The operating system is complicated, as it has to provide realtime processing for the control tasks as well as general data processing.

Display and documents

MICAS does not provide a user friendly interface to a printer or display. For the storage function another operating system would be required as well.

The result is:
MICAS allows access to all data, but has no suitable capability for general data processing, mass storage or data display. To change this not only considerable hardware is required, but a new operating system is needed, that can handle such different tasks as real time processing in periods of tens of milliseconds and event triggered processing as well as general data processing. Beyond this user software has to be developed. There are considerable doubts, that both tasks, real time processing and general data processing can be done with the limited capacity of the MICAS central processing unit.

7.2 Strategy to use MICAS in a data acquisition system

As shown before the task to make MICAS usable for data acquisition is tremendous. Another way to use MICAS in data acquisition is to take advantage of the fact that all data are readily available in the MICAS system and to connect the MICAS system to a general computer which has all the features required for general data processing, storage and display.

This concept is particularly attractive with the availability of small general computers like the PC or other personal computers. These small computers, based on similar central processing units as MICAS do offer systems with all necessary hardware for data storage and display for very low prices, as they are mass products. But the biggest advantage is the availability of system and user software for these computers, which solves all the tasks of data processing including such items as graphic displays of data or generating structured listings in tables, etc.

The use of a PC makes the data processing a task of the end user and he may use what ever PC he likes and do with the acquired data what he wants. To change programs is no problem, as the general software for PC's becomes more and more general knowledge of technical and administrative personnel.

To test this concept it is to study how the MICAS system can be connected to a general computer and which conditions are necessary for the data exchange between MICAS and a PC.

7.3 Concept of a data acquisition system with MICAS and a PC

7.3.1 General capability

The system should provide continuous data about the operation of the plant. The system should provide fault data in the case of a fault which contains different or more detailed information than the continuous data. The concept of splitting up the data acquisition into continuous data and fault data acquisition has two advantages, it unloads the data link, which as a serial link has only limited capacity, and it provides a considerable data reduction already at the source. It should be possible to control the data acquisition from the keyboard of the PC.

The programming of the MICAS system should be done once to set up a structure for the acquisition and communication. This is desired as MICAS programming is considered a special task and requires special training. Nevertheless programming of the MICAS is done as usual for the system by using blocks, for which the user specifies the parameters.

The programming of the PC has two parts. The programs for the communication which store the data in files in the PC and allow a remote control of the MICAS system by starting or stopping the data acquisition and selecting the data to be acquired. The programs to process, store and display the acquired data. All programs are written in general languages for PC's.

7.3.2 Data link

To connect the MICAS with a PC a serial link was chosen. The reason is, that MICAS has a serial link board and that all PC's have some type of serial port. A parallel link would allow higher data rates, but there is none for the MICAS system and furthermore parallel links are much more special and not readily available on PC's in a general standard which prevents the generality desired for the concept.

7.3.3 PC properties and requirements

The PC is a general computer and is not operating in real time. The time it needs for a certain task is dependent on the operating system, the program, the I/O devices involved and the amount of data handled. A task needs typically hundreds of milliseconds or some seconds.

Many PC's have hardware connected to a serial port, which allows to capture incoming data up to a certain amount without interference of the central processing unit. These captured data, called a record, typically of 256 bytes of length are then transferred at once by the program to memory in RAM. A record or several records can be stored to a disk to save the data against loss by power failure or system malfunction. This storing takes some time, as the program overhead is larger than for storing a record and the operation of the disks is involved.

In principle it is possible to interrupt tasks to do other tasks in between, but the programming becomes difficult and if the interruptions happen more often the process becomes very inefficient, due to the overhead.

As a result the operation of the data acquisition system should be such, that the data are sent to the PC in records of a certain length. Between the records must be enough time to allow the PC to store the record. After a set of records time must be provided to allow processing and storage on disk.

7.3.4 MICAS system properties and requirements

The serial port available in the MICAS system is under control of the MICAS operating system. So as not to interfere with this system a second serial port should be installed for the link to the PC. Furthermore the system memory should be extended to have enough intermediate storage to form a set of records to be sent to the PC, and to provide intermediate storage space for the fault data, which may be collected so fast, that it exceeds the sending rate in the serial channel. In many cases an extension by one memory board is sufficient. MICAS programming blocks have to be developed which gather the data and allow the communication with the PC.

7.3.5 Principle function

A program called CODA, continuous data, in the MICAS collects at a certain sample rate data from nc channels. It packages these data in a set of records and supervises the sending of the set of records to the PC. This program is executed periodically, as are all other MICAS blocks. This program takes care of the sending pauses between the records and the pauses between the sets of records, so that the PC has enough time to do its tasks.

A program called FADA, fault data, in the MICAS collects at a fast sample rate, normally the smallest timing unit of the control program, data from fc fault data channels. These channels may or may not overlap with the continuous channels. The fault data are written to consecutive memory locations until the foreseen memory is filled up, then the oldest data are overwritten. This continuous writing of data ends in the case of a fault within a predetermined time after the fault occurred, thus providing data before and after the fault. The data generated by FADA in continuous writing are not send to the PC. Only in the event of a fault FADA sets a flag and then the fault data can be sent down to the PC under control of the program CODA, instead of the continues data. FADA is executed periodically.

A program SERE (send receive) which is interrupt driven, does the actual sending and receiving of bytes from the PC. This program has a very high priority so that it can use the capacity of the data channel. This program constitutes the main loading of the MICAS system by the data acquisition system.

Two other programs AUX1 and AUX2 are initialization and decoding programs, which allow the configuration of the data acquisition by commands send from the PC.

The PC is programmed in FORTRAN and has commands to start the data acquisition, to terminate it and to get the fault data, if a fault is signalled. It is possible to select by keyboard commands on the PC the channels for continuous and fault data recording and to select the sample rate for the continuous data recording.

7.4 Verification of the MICAS PC link for data acquisition

To test the concept the MICAS system was connected through a serial link to a PDP 11 general computer. The details of this set up are described in the following sections. They contain the developed program blocks and provide the timing and loading analysis required for the communication system.

7.4.1 Timing structure, communication and processing capacity

As outlined before, the use of a general computer with general software requires, that the data are packed in records and send to the computer by the MICAS system. Between these records there has to be enough time to store the data to the disk of the computer and for processing.

For this the data are accumulated and then send as quickly as possible in intermittent records. This technique requires that x bytes of data can be transmitted in less time it requires to sample x bytes. The time difference between the two is free time and can be used by the receiving system to process and store the received data. By accumulating several records of data, and then sending them all in quick succession, the free time for processing may be increased.

Figure 7.1 shows the principle timing diagram for sampling and sending. The MICAS system accumulates l records of data. It requires μ seconds to sample and store each record. After y sec. the process repeats itself and another l records are obtained.

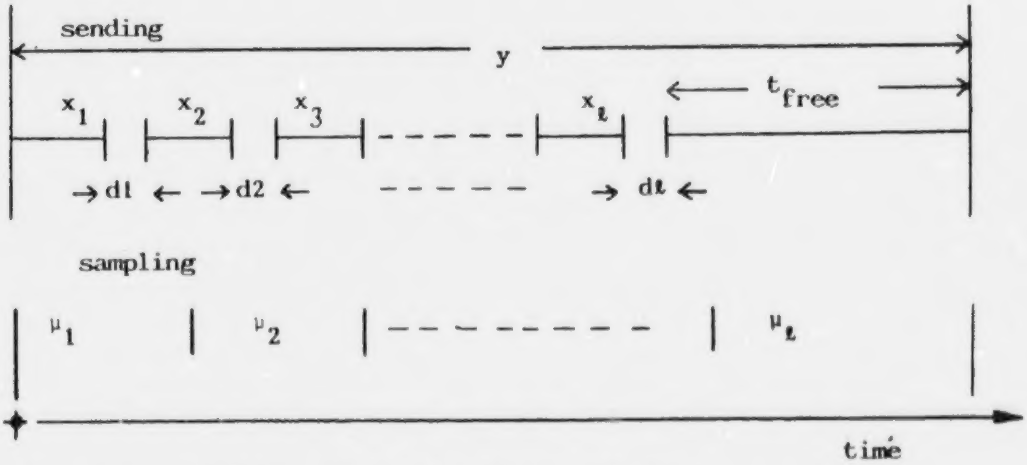


Fig. 7.1 Sampling and Sending Timing Diagram

Once l records have been stored, the process of transmitting these data is initiated simultaneously to the start of the next record accumulation. Depending on the Baud rate, selected for sending, it takes x seconds to transmit each record. A delay d is added between transmitting each record. This delay will be a function of the minimum time between successive reads at the receiving end. The total free time available to the receiving system is then:

$$\begin{aligned} t_{free} &= l\mu - (ex + (l - 1)d) \\ &= l(m - x - d) + d \end{aligned}$$

One important system dependent parameter is b, the size of each record. A record can further be defined as the largest number of bytes that can be received without interruption at the receiving end. It is assumed that each record will be transmitted at a maximum rate by the MICAS system.

For the μPDP-11 used as general computer and for most personal computers using FORTRAN, this parameter is determined by the largest record that may be input by a single read statement; it is 255 bytes. Thus for this particular system $b_{max} = 255$.

Let

n = the maximum number of data channels that are to be transmitted

t_s = the sample period of the accumulating system (i.e. data is sampled and stored every t_s seconds)

K = integer multiplier so that actual sampled period = Kt_s

Assuming each byte of data is converted to tow ASCII bytes for transmission then μ, the time to sample and store each record is:

$$\mu = Int\ (b/2n) \times K \times t_s$$

To calculate x, the time required to transmit one record of data, and again assuming a continuous fast interrupt driven transmission of each record by the MICAS system, t_x is defined as the minimum time between the transmission of any two bytes.

therefore $x = b \times t_x$

Note:

t_x for 9600 baud ≈ 1 ms

for 4800 baud ≈ 2 ms etc.

The other system dependent parameter is d, the delay time between the sending of records. In a receiving system where each record is a single read, the minimum delay is the time between the two successive reads, approximately 30 ms on μPDP-11 with FORTRAN. A practical delay will also have to allow for some processing however. This usually includes the temporary storage of each record in an array and tests to determine whether a valid data record has been received rather than some other information.

l is a parameter which in theory may be fairly large. The only restriction on l is the size of temporary storage (read/write memory) than can be utilized, both at the send and at the receiving system. Clearly, though, the larger l is the more free time there will be available. In the system implemented for μPDP-11, $l = 4$ which provided $4 \times 250 = 1K$ of transmitted data each cycle.

The following parameters were chosen for the test with the MICAS system connected to the μPDP11:

b was chosen to be 250 bytes for convenience (≤ 255)

n = 4 channels, $l = 4$

t_s = 50 ms sample period

d = 100 ms to allow ample head room

K =1 increasing K increases available free time

Therefore

$$\mu = Int\ (\frac{b}{2n})\ Kt_s$$

$$= Int\ (\frac{250}{8})\ (50ms)$$

$$= (31)\ (50ms) = 1.55$$

$$x = b \times tx$$

With a Baud rate of 4800, chosen to reduce the loading of the MICAS system, it is :

$$x = (250)\ (2ms) = 500ms$$

(2 ms is based on 4800 baud)

Therefore

$$t_{free} = l\ (\mu - x - d) + d$$

$$= 4\ (1.55 - 0.5 - 0.1) + 0.1$$

$$+ 3.9s$$

This 3.9 seconds are available to open a file, process and append the new data to the file, and close the file, which includes a write to the disk.

A plausible configuration would see $l = 40$. This would require $40 \times 250 = 10K$ bytes read/write storage in each system, but would allow a processing time of

$$t_{free} = 40\ (1.55 - 0.5 - 0.1) + 0.1$$

$$= 38.1s$$

7.4.2 MICAS system loading analysis

As discussed before, the requirements of the use of a general computer requires a certain structure and sequence. It is desirable to use the highest

baud rate available with the given equipment, to gain time for processing. The MICAS system has to provide the data and therefore the sending was given the highest priority to utilize the capacity of the serial link and to reach the desired high transmission speed.
In principle this should not result in a high loading of the MICAS, but in the given operating system interrupts use a considerable overhead. The following load analysis for the MICAS system demonstrates, that this is the limiting factor to use the full speed of the serial link of 9600 Baud.

Note: Note: the load A for a cyclical Block Set is defined as:

$$A = \frac{\varepsilon \; running \;\; times + 0.4ms}{cycle \;\; time}$$

It is further inferred that an event (interrupt) block's running time be added on to the Block Set's running time. By experimental, it was found that another 0.4 ms must be added for each additional event triggered block.

TRM (existing software)

The TRM MICAS program consists of 4 block sets with cadculated execution times as follows:

Block Set	**Start Address**	**Label**	**Cycle Time**	ε Running Time
BSA	4100	F001	10ms	2.2ms
BSB	500	F300	20ms	1.0ms
BSC	4900	F600	30ms	2.5ms
BSD	5800	FA00	40ms	0.79ms

Then

$$A = \frac{2.2 + 0.4}{10} = 26\%$$

$$B = \frac{1.0 + 0.4}{20} = 7.0\%$$

$$C = \frac{2.5 + 0.4}{30} = 9.67\%$$

$$D = \frac{0.79 + 0.4}{40} = 2.98\%$$

Total loading = 45.65%

Block Set	Type	Start Addr	Label	Cycle Time	Running Time
CIOINIT	INIT	4080	FE00	-	-
CIOINT_Y5	EVENT	40C0	FE30	-	R-49μsec, X-88μsec, RX-106μsec*
A	CYCLE	4100	F001	10ms	2.2ms
B	CYCLE	4500	F300	20ms	1.0ms
CIOFDS	CYCLE	4700	FE50	20ms	(120n+142)μsec**
C	CYCLE	4900	F600	30ms	2.5ms
D	CYCLE	5800	FA00	40ms	0.79ms
CIOSYS	CYCLE	5900	FE70	50ms	(120n+338)μsec**
CIOBGR	CYCLE	5980	FE90	520ms	20ms***

* R-receiver interrupt, RX-both serviced in one execution
X-transmitter interrupt

** based on worst case
n=no. of data channels

*** The dominant time delays in CIOBGR occur if waiting for incoming characters is necessary. The longest command is 10 characters, therefore at 4800 baud, running time ≈ $10 \times 2ms + 20ms$.

Load Calculations

The effects of CIOINT_V5 are taken into account in the calculation of the fastest cycle time loading. The fastest cycle time is 10ms. At 4800 baud, a transmitter interrupt can occur at maximum of once every 2.083ms. There is 0.4ms overhead for the 10ms block set. Thus in 10ms, there can only be 4 interrupts. (Note: In theory, there could also be receiver interrupts during a 10ms period. However, the design of the system is such that transmitted data and received data are always mutually exclusive.)

Therefore

$$T_A = \frac{2.2 + 0.4 + 4(.088 + 0.4)}{10} = 45.5\%$$

For $n = 4$, the running time of CIOFDS

$$= (120(4) + 142)\mu s = 622\mu s$$

$$T_B = \frac{1.0 + 0.4}{20} = 7.0\%$$

$$T_{TFS} = \frac{.0622 + 0.4}{20} = 5.1\%$$

$$T_C = \frac{2.5 + 0.4}{30} = 9.67\%$$

$$T_D = \frac{0.79 + 0.4}{40} = 2.98\%$$

For $n = 4$ the running time of CIOSYS

$$= (120(4) + 338)\mu s = 818\mu s$$

$$T_{SYS} = \frac{.818 + 0.4}{50} = 2.490$$

$$T_{BGR} = \frac{20 + 0.4}{520} = 3.9\%$$

Therefore

$$Total\, loading = 76.55\%$$

Notice that the primary increase in loading is due to the interrupt routine specifically its frequency interruption, and the overload it carries (0.4ms function of MICAS).

Notice also that the frequency of interruption depends on baud rate and the sources of the interrupt. For example, the original design used a baud rate of 9600 which could mean up to 9 interrupts in a 10ms period.

In that case T_A = 70% alone. This causes a total loading greater than 100%.

Finally, observe that to go to an 8-channel system (8 channel continuous and 8 channel fault) only increases the total loading by 3.4% to 80%. As a result of the loading analysis the transmission baud rate, originally set at 9600 was reduced to 4800.

7.5 The MICAS communications operating system

MCOS3 is the collection of programs residing at the MICAS system of the MICAS-μPDP-11 communications link. It is designed to run concurrent and transparent to any existing application programs running on MICAS system. Testing of the communication system was performed while the control programs were running. Of course loading of the system must be taken into account.

As mentioned earlier two additional MICAS boards are required: a second serial port card (10UP03) and a second memory card (10CP01). The serial port provides a dedicated serial link to the receiving system. The memory card provides storage space for

(1) the program blocks in (2 * 2K EPROMS) = 4K (only partially filled)

(2) intermediate storage for passing parameters, etc.

(3) data buffer areas for accepting incoming commands and accumulating outgoing data. (the remaining 12K)

Only one hardware modification is necessary, namely the connection of the serial port's interrupt request line to an event triggered input on the timer board. (18,77f6 to 18,25z6)

MCOS3 consists of 5 MICAS type program blocks. Each of these blocks resembles any other MICAS block in that they have a call address followed by a list of address specifiers. Each is program location independent and can be inserted anywhere in the MICAS program environment. They differ from a standard MICAS block in that they must all (but one) be active for MCOS3 to function correctly. Also, there is a conceptual difference. While MICAS blocks have address nodes to connect outputs to other inputs, MCOS3 blocks are usually passing parameters, status information or variables through these addresses. Finally these blocks access an absolute memory area (on the second memory card) as well as general intermediate memory.

The five program blocks of MCOS# are described in the following paragraphs. The Block specifications and the assembler listing of this newly developed MICAS blocks and flow charts to support the readability of these programs are provided in the appendix A6.

7.5.1 The program block AUX1 (CIOINIT) and AUX2 (CIOBGR)

The block AUX1 is executed as an initialization block and thus is run only once upon start-up. Its main purpose is to configure the command register of the auxiliary serial port. It also sends a title to the terminal.

The block AUX2 is executed as a cyclical block with a cycle time of 520 ms. Its function, which is to interrupt valid commands received from the PDP-11 (or other data acquisition system), is of a low priority - thus the slow cycle time. As characters are received into the receiver buffer memory area, this routine passes the character strings and isolates any valid commands. These commands (see command table) are then executed by assigning values to the appropriate variables. Nine variables are passed by AUX2, as well pointers to each continuous data and fault data address.

7.5.2 The program block SERE (CIOINT_V5)

This block is executed as an interrupt routine or 'event triggered block'. It is given the highest priority in the MICAS system execution structure. It is controlled by the auxiliary serial port interrupt request line and thus is executed whenever (a) a character has been received or (b) whenever the transmitter is enabled and is ready to transmit a character. The interrupt routine identifies which, if any, of these two cases is true. For a received character, the character is stored in the receiver buffer memory area. For a transmit ready case, a character is removed from the transmit buffer memory area and sent to the serial port. Two variables are passed by SERE: Dreq and XEN. Both are flags and thus can be true or false. Dreq or, 'Delay request' is set true when a carriage return is transmitted indicating the end of a transmitted record. CODA uses this to implement a delay system between records. XEN or 'Xmitter ENable' indicates whether the transmitter is enabled (true) or disabled (false), which is also used by CODA.

7.5.3 The program block CODA (CIOSYS)

This block is executed as a cyclical block with the chosen cycle time dictated by the required sample rate for continuous data acquisition. For testing a 4 channel system the cycle time or sample time was 50 ms. There is no specific restriction on what speed this block runs out. However system loading must be considered. Note also that the execution time of this block increases proportionally as the maximum specified number of channels increases. CODA is the main system block of MCOS3. The memory and buffer structure are set up and maintained by this routine. Most of the commands that AUX2 executes act on CODA. This routine is also responsible for all the asynchronous transmission of data to the remote system (by monitoring SERE). Finally CODA acquires, processes and stores all data for the 'continuous' acquisition system. During 'continuous' operation, then, this block is simultaneously acquiring continuous data and supervising the sending of the same data. During 'fault download mode', this routine is only supervising the data transmission. FADA is responsible for the acquiring of the "fault" data. CIOSYS passes and receives seven variables, six constants and pointers to the 'continuous' data addresses.

7.5.4 The program block FADA (CIOFDS)

This block is executed as a cyclical block with the chosen cycle time dictated by the required sample rate for 'fault data acquisition'. For a 4 channel fault channel system, the cycle time or sample time was 20 ms. There is no specific restriction on what speed this block runs at either. It is usually desired that this rate be as fast as possible. System loading must be considered as well as increase execution time due to increased number of channels. CFADA is responsible for all fault data acquisition. It acquires, processes, and stores data in a buffer (up to 10K bytes) which is continuously overwritten when a fault condition occurs a STOP flag becomes true, and after a predetermined delay, data acquisition ceases. During a download, data in the fault data buffer is transmitted under the supervision of CODA. Nine variables and four constants are passed, as well as the pointers for the fault data addresses.

7.5.5 The μPDP11 program DCOS5

To do a full verification of the proposed concept a program for the general computer was written. This program demonstrates some of the possibilities of controlling the data acquisition. The program is shortly described below. The program listing with flow charts is attached in appendix A6.

The DCOS5 program allows the user to acquire selected data from the MICAS controller, while communicating entirely from the remote PC computer. The program essentially consists of an interactive mode whereby the user is queried for set up information and two acquisition modes: one for continuous data acquisition. Upon exiting from any mode of operation, the keyboard is polled awaiting the next command request to be entered.

Interactive Mode (Key 'I')

The following queries are made to the user:

1. no. of channels for continuous acquisition
2. MICAS address assigned to each cahannel
3. sample rate desired (20K ms, k=1,2...)
4. no. of channels for fault acquisition
5. MICAS address assigned to each channel
6. delay time in no. of sample cycles that fault storage is to continue after a fault has occurred.

Continuous Data Acquisition Mode (Key 'R')
(referred to as Block R in flowchart)

Upon entering this mode, the following takes place:

1. A data disk file is opened (dcos5.ud1).
2. A command is sent to MICAS which starts continuous data transmission at the MICAS end.
3. A record is read at the serial port. The record is interpreted as either data or "handshaking information". If it is the latter, the appropriate action is taken. If it is data, the data is processed and decoded and written into the opened file.
4. The data file is then closed which causes it to be written to the hard-disk.

5. The file is then re-opened.
6. Provided there is no request to stop, the next record is read (go to step (3)).
7. If during the acquisition process, a fault signal is received, a flag is set for future reference and a message is sent to the user terminal indicating that fault data is now available.
8. The user may halt continuous data acquisition at any time by typing 's' on the keyboard.

Fault Data Acquisition Mode (Key 'D')

Once a fault signal has been received, they enter this mode to acquire the fault data. The following then occurs:

1. A separate disk data file (dcos5.ud2) is opened.
2. A command is sent to MICAS to start transmission of fault data.
3. All the fault data is read from the serial port, processed, and stored to the data file.
4. The data file is closed, causing a write to the hard disk.
5. Control returns to polling the keyboard for the next request.
6. The fault storage routine (which has been halted due to a fault) may be reset by typing 'c' on the keyboard. This resumes MICAS fault storage.

Processing of Data

Once the data has been stored in a file, it can be displayed, graphed, or copied to a floppy disk and analysed on another system. The commands 'n' and 'p' will display in tabular form the data files for continuous and fault data, respectively.

Additionally, a third file called dco55.p1m is created after an interactive session. This file contains information pertaining to the sample rate of acquisition, the number of records stored, etc. This file along with one of the data files can then be accessed by another program called "Graph" which will display a plot versus time of the acquired data. Of course, other such utilities may be written to further analyze the acquired data.

7.6 Results

The concept is feasible and the data rates achieved seems sufficient for many applications. An improvement of the data rate is possible by a CODA program, which would allow different sample rates for different channels, such providing a data reduction at the source. In this case the data rate available with a serial port should no longer be a real restriction.

The main restriction at the moment is the loading of the MICAS system by the program SERE, which delivers the single bytes to the serial port for sending. The program has to run for each byte and therefore interrupts often, this could be possible, i0 not for the overhead of an interrupt in the MICAS would be so severe. The result is, that to keep the loading for the data acquisition in

limits, the possible data rate of the channel cannot be used.
The solution is possiblely another board, which contains a serial port and an intermediate buffer for one record and which can send one record without the need to call the central processing unit.

8. Diesel and locomotive speed controls and their impact on fuel usage

The control of a diesel locomotive is done by the throttle. The setting of the throttle gives a traction power command to the electric drive and a speed command to the diesel engine. The resulting operating point of the diesel engine is determined by the traction power required by the electric drive plus the power for the auxiliaries and the set diesel engine speed. The selection of the diesel engine speed for these operating points is such, that the diesel operates with high fuel efficiency. Typically the operating points for notch 5 to 8 are all on or close to the line of constant specific fuel consumption [5],[6]. These principle situation is shown in fig. 8.1.

8.1 Potential for fuel savings by controls

As long as the locomotive is operating in these operating points, there is no potential to improve the fuel efficiency by a change of the diesel engine speed. Only if the diesel engine does not operate at its full power, i.e. for notch 7 at the point A1 or A2 in fig. 8.1, it operates below its best fuel efficiency. In this case a better efficiency would be achieved, for an operation at a lower diesel speed, i.e. nA1 for the power at A1 or n6 for the power in A2, which is operation in notch 6.

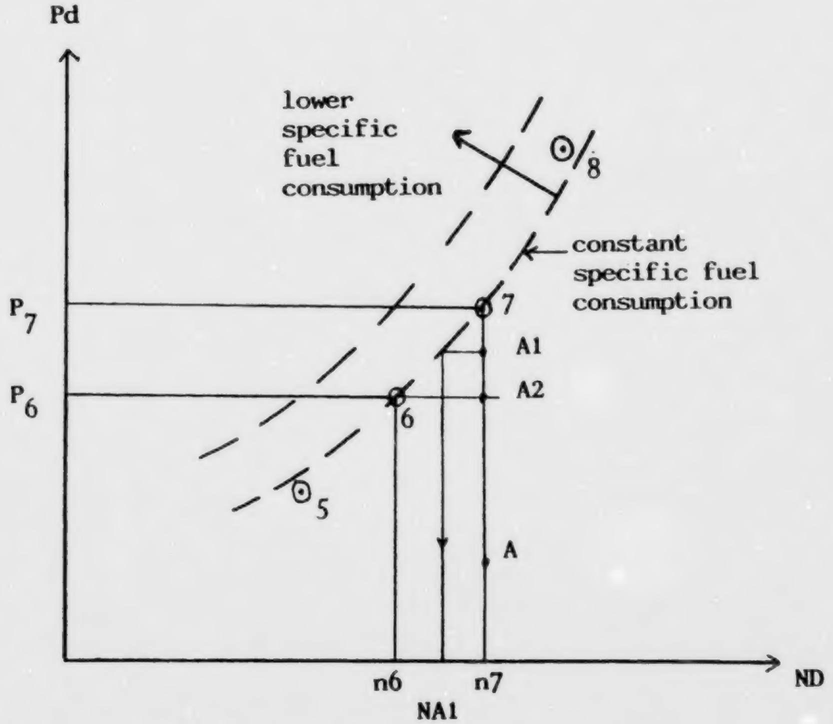

Pd
lower specific fuel consumption
8
constant specific fuel consumption
P_7
7
A1
P_6
A2
6
5
A
n6
n7
NA1
ND

Fig. 8.1 Fuel Efficiency Control

Operation at lower than the full power occurs in two situations. If the auxiliary power is not fully needed and if the locomotive operates under locomotive

speed control.
Not fully used auxiliary power has only minimal effects on the specific fuel consumption in higher notches. The influence is larger in lower notches, but a main line locomotive operates so little in these points and the absolute fuel savings are so small, that the influence on the fuel consumption can be ignored.

If the locomotive operates under speed control there are two possibilities to control the speed.
The speed control influences the notch position. In this case the control switches between notches to adjust the speed. A type of bang bang control results, as the steps in power between the higher notches are large. The operation at a notch setting is at optimum specific fuel use, but it is questionable, whether the switching between notches does not lower the efficiency. The switching frequency depends on the accuracy of the speed control.
A better speed control results, if the speed control influences the electric drive. A very accurate control results. But in this case the operating point is mostly below the optimal specific fuel use, and a decrease of the diesel speed would improve specific fuel use. Under the the assumption, that the diesel speed control is not altered, a decrease in diesel speed is possible only in steps of the notches.
A fuel efficiency control has to monitor the actual power used, and if it is possible to operate at a lower notch, to select this notch to increase fuel efficiency. To avoid frequent switching between notches, if the power operating point is close to a notch (i.e. point A2 in fig. 8.1), a hysteresis is used. This hysteresis does not effect the accuracy of the speed control.

The proposed fuel efficiency control will avoid the operation in an unnecessary high notch, but as the step between notch 7 and 8 is very large it will operate in most cases in notch 8. But there is a possibility in multiple traction to achieve a better use of the fuel efficiency control, if not all locomotive are operated at the same notch. This provides a much finer control of the fuel efficiency, as a change at one locomotive is only a small step in power.
As the ac locomotive has a notch control for the braking in the MICAS controller, this control could be used in a fuel efficiency control for multiple traction with locomotive speed control.

8.2 Fuel efficiency control for multiple traction with speed control

A potential to safe fuel by diesel speed control is seen only for multiple traction with locomotive speed control, as outlined before. The principle idea is not to operate all locomotives in the same notch, but to use one locomotive as a vernier and change the throttle command to improve fuel efficiency. As outlined earlier the operation at higher throttle operating points is normally at high fuel efficiency.
An example may demonstrate the situation. Three locomotives with 3 × 100% = 300% power in notch 8 operate together. Notch 8 is 100% power, notch 7 is 70%, notch 6 is 50% and notch 5 is 34% power. If the speed control asks for 230% power all locomotives operating in notch 8 would be not fully loaded and operate at a higher specific fuel consumption. To operate all at notch 7 does not cover the required power. But operating two at notch 8 and one at notch 5 provides maximal 234% power and operates all three locomotives close to the full notch power, that means at a low specific fuel consumption.

The required power can be sensed from the traction power command given by the locomotive speed control to the electric drive of the vernier locomotive. A control algorithm adjusts then the throttle of the vernier locomotive such, that all locomotives operate close to the maximum notch power.

To define the fuel saving potential of this proposed control, a theoretical study is carried out and the fuel savings are calculated under certain assumptions.

8.3 Theoretical fuel savings

To do the calculations, assumptions about the specific fuel efficiency and the throttle operating points have to be made. The throttle operating points were taken from the locomotive used in the simulation. The lines of constant specific fuel efficiency are assumed based on similar curves in [5], and the earlier mentioned source [7], which indicates that the operating points in higher notches are close to the same specific fuel consumption.
Figure 8.2 shows the resulting assumption in a graph of the diesel power versus the diesel speed. The values are shown in the same per unit values used for the simulation. A series of lines of constant specific fuel use is drawn onto the diagram.

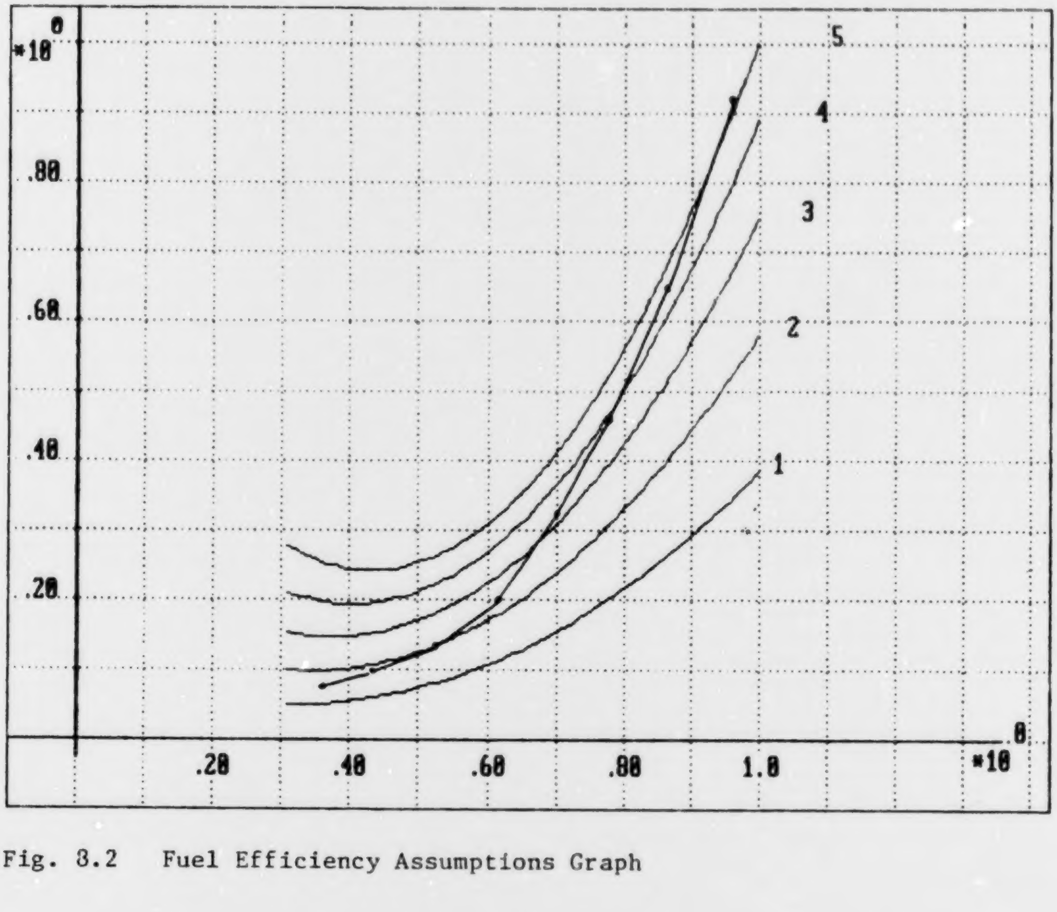


Fig. 8.2 Fuel Efficiency Assumptions Graph

The equation describing these lines is

$$Pd = (0.30 + 52/6 \times z) \times (nd - (0.29 + 0.6 \times z))^2 + z$$

The specific fuel use is

$$fs = (1.166 - 0.69 \times z) \times 214 g/kWh$$

The parameter z varies from 0.048 to 0.24 in steps of 0.048 for the lines of constant specific fuel efficiency shown in fig. 8.2.

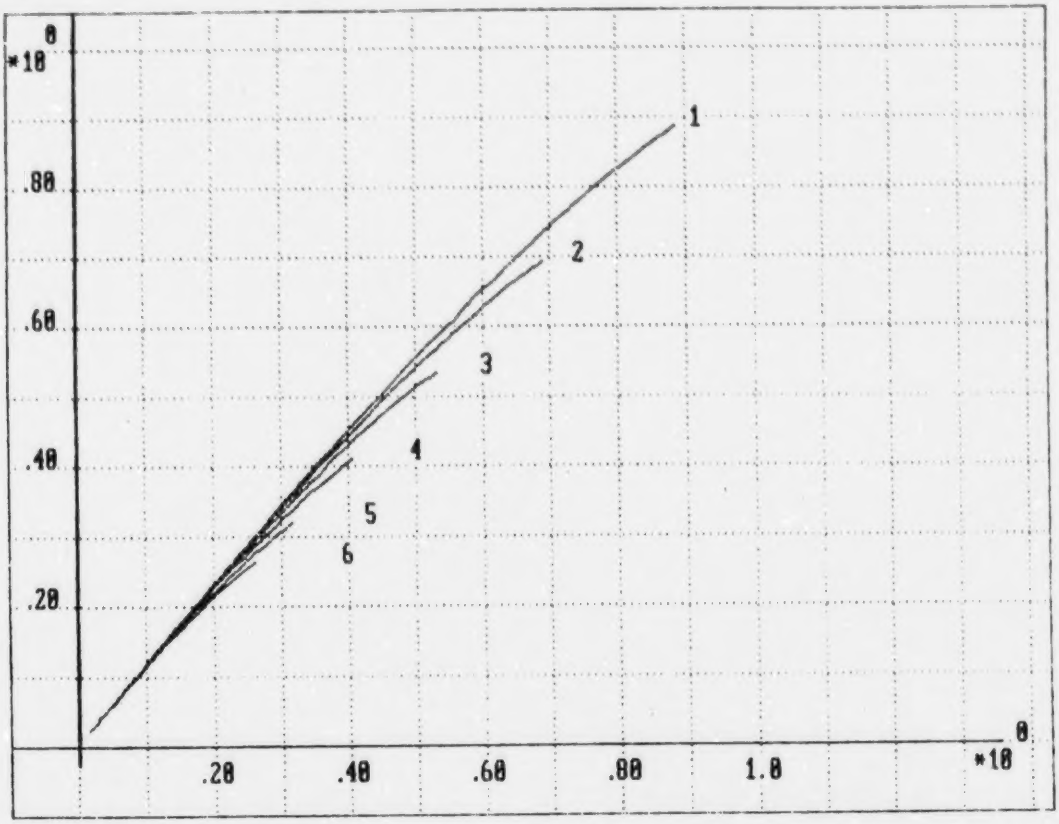


Fig. 8.3 Fuel Usage vs Power

Based on the above assumptions the fuel usage for a single locomotive with speed control is calculated. Figure 8.3 shows the fuel usage in function of the power for the operation in different notches. The fuel use is shown in per unit values, with the base value of 785 kg/h. It clearly can be seen, that selecting a lower notch, if the power can be provided in this lower notch, reduces the total fuel consumption.

The calculation of the fuel savings for multiple traction with one locomotive used as vernier for the speed control, is based on three identical locomotives, with the fuel use characteristics as assumed above. The fuel consumption for four cases is calculated. The results are shown in fig. 8.4.

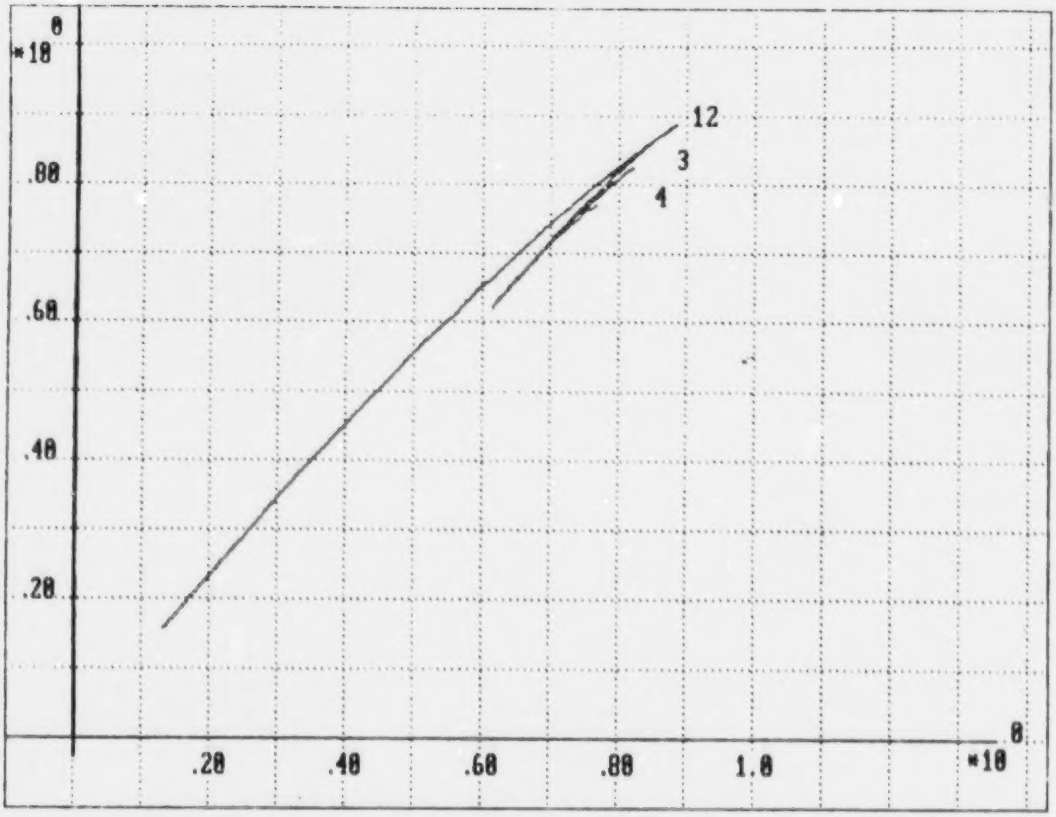


Fig. 8.4 Fuel Usage - Multiple Traction

The first case is the operation without using one locomotive as a vernier. In this case all locomotives operate in notch 8 and if the full power is not required to maintain the speed, all 3 locomotives operate equally below the maximum power point of the notch. In the three other cases only one locomotive is used to control the speed, the two others operate at the maximum power in notch 8. In case 2 the speed controlling locomotive operates in notch 8. Only this locomotive operates at reduced power, if the power is not required to maintain the speed. The cases 3 and 4 show the fuel consumption, if it is possible to operate the speed controlling locomotive in notch 7 or even in notch 6.

It can be seen, that the fuel consumption is lower, operating only one locomotive below the maximum power. Switching this locomotive to lower notches increases the fuel savings. Figure 8.5 shows the absolute fuel savings in percent as function of the required total power. The savings are with reference to an operation as in case one, where all locomotives are equally reduced in power output.

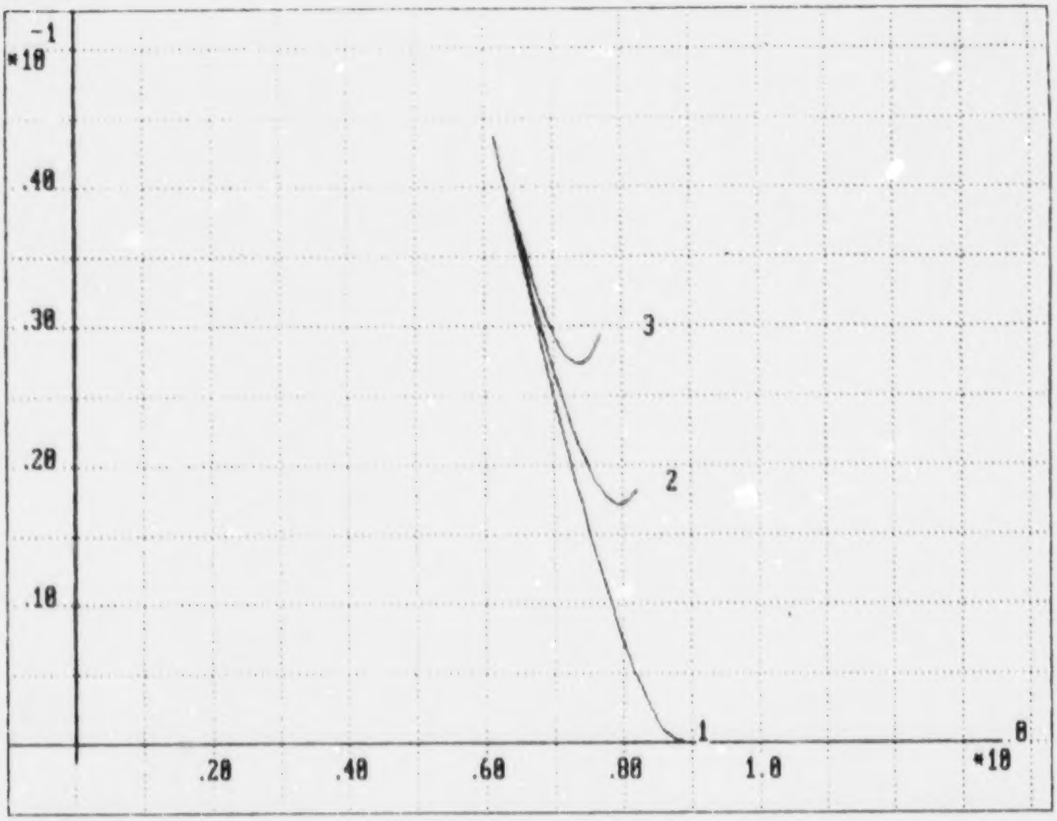


Fig. 8.5 Fuel Saving vs Required Total Power

The calculations show, that the potential for fuel savings is in the order of 2% to 4%, if the required power for the speed controlled operation is between 90% and 70% of the maximum available power.

8.4 Fuel saving by train control

From the discussion before it is obvious, that there is no real potential for fuel saving by control, if the locomotive is operated at the full maximum power of a notch. The potential fuel saving for locomotives operating with speed control, does not apply to many cases, as most locomotives in North America operate with power control.

The discussion so far was based on a simple single mass train model. This allows to discuss fuel economy as fuel economy of the locomotive with the traction diagram and the setting of the throttle as the important influences. But a train is a very complex system, and it is well known, that a train is operated not only by the locomotive power, but that the use of the brakes is very important for the operation. This is particular true for the control of the stability of the train and the control of the forces in the train especially in the couplers.

It is well known, that the coordination of power and brake application has an impact on fuel consumption. It seems, that this coordination is the major factor in the function of some fuel saver system in the market. They prevent, i.e.

extended brake application together with power application. But to achieve optimal operation much more than suppressing extended braking is required.

To achieve fuel saving by train control a complex control system is needed. This system may be called an expert system for train handling. It will not only optimize fuel usage, but also improve the transport system and lower stress on the equipment. Such a system should be studied, as it is felt, that there is the major potential for fuel saving by a control system.
Such a study seems particular interesting, as it seems possible to design such a system with today's available microelectronic and it could be part of the planned advanced train control systems. Some major components for the control system are now available or will be available in the near future. These are powerful on board computers, rugged enough for train operation, locomotive control systems, as proposed in this project, economical systems to determine the train location along the track [7], knowledge about modeling train dynamics with small computers [8] and an emerging theory about expert systems.

9. Laboratory test by BBC using an AC traction motor and an inverter

9.1 Introduction

This report describes the setup and testing of the microprocessor test rig. The purpose of this rig is to permit testing of experimental control software in an environment approximating the locomotive. It consists of the MICAS computer inverter control, inverter and motor. The rig is intended to evaluate the operational characteristics of new variations of control configuration in a safe, controlled low power environment.

The standard control configuration is a constant torque demand scheme wherein the inverter control is fed a calculated tractive effort demand signal. The control then supplies a leading frequency relative to the instantaneous motor speed to produce the torque demanded. The motor speed and hence frequency are a closed loop function controlled by the physical interaction of the wheel to rail, slip, trailing mass and gradient. An alternative control configuration would be to control frequency in an open loop mode which would yield absolute wheel speed and hence wheelslip control.

The application of such new control functions can create unforeseen reaction situations which, if directly applied to a high-powered vehicle, could prove dangerous to both personnel and equipment. The test rig allows operation and debugging in an environment that is safe for both personnel and equipment.

9.2 Equipment

The equipment consists of the following:

1 - Traction Motor

Model	4077
Make	BBC
Type	3 phase asynchronous
Pole	6

1 - Inverter

Model	13SG07A
Make	BBC
Type	voltage source
Rating	volts (AC) 780
	current A 715
	power KVA 966
	freq. range Hz 0-150

1 - Rectifier

Make	BBC
Type	3 phase full wave bridge
Rating	volts 1500
	amps 150

1 - Control Set

MICAS computer

Inverter control and supervision

Power supplies

1 - Limit Resistor

Rating	8.3 ohm
Amps	100

1 - MICAS Service Device

The test rig was designed in a modular fashion to permit ease of configuration as well as ready portability.

Figure 1-A shows a general view of the test set-up. The motor which is shown in figure 1-B is to the extreme right in the general view. It is directly connected to the output of the inverter. The rotor is free with no mechanical load. The inverter and rectifier, Figs. 1-C and 1-D are installed as a unit. The rectifier is mounted on the right hand side of the inverter when looking at the front of the inverter. The inverter is comprised of the filter and commutating capacitors mounted in the lower half. The upper half contains the thyristors, diodes, and firing transformers in plug-in modules. The upper three modules are the positive switching devices, the lower three the negative. The three center modules are the commutation devices. The vertical stacks, each composed of a positive, negative and commutating module are the switching sections for each of the three phases. On the top of the inverter are located two LEM 400 amp isolated current sensors, one in each of two phases.

The last component on the power side of the rig is the curent limit resistor, Fig. 1-F. In a normal locomotive application, the power would be delivered to the inverter via a dedicated alternator and rectifier. The inverter control used in traction applications is designed, in the event of a commutation failure, to turn all thyristors on. This creates a massive short circuit, however, since the alternator excitation is designed to limit the output current to a preset value, and the current is distributed through all devices. The value of the current in any device is less than its maximum acceptable level. The alternator output would then be reduced to zero and the inverter and control reset and normal operation recovered.

In the test rig, the power is derived from the building utility supply. The utility is a low impedance system as opposed to the alternator regulator which, with its defined current limit, is a high impedance system. If the inverter were to enter its protection mode and fire all thyristors, the only limit on curent would be the circuit impedances and ultimately the operation of protective fuses and circuit breakers. While this protection is adequate for random low frequency events, for a test rig where such an event will be induced, such protection would be both expensive and risk prone. During early testing, a protective firing resulted in the destruction of the rectifier bridge due to high transient voltage resulting from the operation of protective devices.

In order to simulate the system on the locomotive, it was necessary to add a series resistance in the dc link between the rectifier and inverter. The resistor was sized such that in the event of a protective firing, it would limit the current to a level sustainable by the building power supply.

Further, it had to be of a low enough value to permit adequate voltage at the inverter input for proper commutation at the operating power level. In order to achieve this compromise, an 8.3 ohm resistor bank capable of sustaining a 100 amp current was chosen.

The control system was designed to be a self-contained unit. Built in a mobile rack assembly, the system requires only a source of 100 V ac power to be functional, Figs. 1-A and 1-E. The system incorporates the complete control system of a locomotive. Referring to Fig. 1-E, the left hand rack from the top down. The top two racks contain the system-regulated power supplies. The two lower racks contain the inverter firing and protection circuits. The front plugs contain the input and output connections to external devices.

The system is electrically divided into three zones. The outer zone is the connection to external devices and systems. These are the signals at high potentials, up to 74 V dc, that are subject to high levels of transient voltage. Connections to this zone are made through the front plugs of card connections. The signals are processed through isolators before being transmitted from or received by the second level computer buss. This zone is characterized by its 15 volt logic with its high noise immunity. These signals are used for inter-rack communication and to and from the input/output isolators. The third or inner zone is the 5 volt CPU and memory level. The signals on this level are restricted to the local rack only. This is the most noise-immune level of the system. The three levels are each served by isolated power suppliers to further ensure signal integrity.

The computers operate from programs stored in read only memory. The computer employs a hardware reset function independent of the CPU to restart the system in the event of a computer glitch.

Connections between the control rack and the inverter are made via a terminal board on the end of the cart. Wiring designations are the same as on the locomotive.

A control box connected to the terminal board permits simulation of all the vehicle's operator controls. The last item in the system is the MICAS service device. This portable instrument is itself a MICAS computer with special software for communication and programming. In use, the program ROM of the target computer is placed in the service device and a memory card from the service device is placed into the target computer. An RS232 link is made between the two computers and a jumper plugged into the target system which vectors its operation to a test program. In order to permit manipulation of the program, it is transferred from Rom to Ram (Random access memory) in the service device, then to Ram in the target system. The process of operating the program in Ram permits rapid testing and modification of the program. Once a suitable version of the software has been developed and tested, a permanent copy can be programmed on Rom and placed into the target system for normal operations.

The language used is the MICAS language which permits programming in logic or analog circuit element blocks directly from a schematic. Modifications require only that changes be made to the device or connection list to change the functioning of the systems. No physical changes are required.

9.3 Test results

9.3.1 Configuration

Figure 2 is a block diagram of the system components and their interconnection. The primary power source was a 200 amp fused building primary feeder. This was connected to a 100 amp breaker located adjacent to the test rig. The breaker permitted local control of the supply current for both normal and abnormal operating conditions such as a protective firing. The output of the breaker is connected to the rectifier input. The output of the rectifier is connected on the positive side to the limit resistor and on the negative side directly to the inverter intput.

The output of the limit resistor was connected to the positive input of the inverter. The output of the inverter was directly connected to the external load; in this case, the traction motor.

The control of the inverter is effected by connections to the module face connectors Fig. 1-C. These connections convey the firing pulses to the thyristors. Additionally, the di/dt current versus time signals used to monitor the operation of the inverter are fed back to the control. These signals yield the high speed data necessary to detect a commutation failure and initiate a protective firing.

The final element contained in the interconnections is the integrity detection safety loop. This feature ensures that all the inverter elements are properly installed and connected. The inverter control is connected to the MICAS computers by back plane connectors. The MICAS computer connects to the control console via the terminal boards and card front connectors. The MICAS service device connects to the computer via a dedicated RS232 link.

9.3.2 Power Supply Test

The object of this test was to verify that the power supply as configured would be adequate both from an operational and a safety standpoint. The first step was to characterize the power supply at the rectifier terminals. This was achieved by measuring V1 at no load, with the 8.3 ohm resistor, R LIMIT, across the rectifier output and with 2 times R LIMIT across the output as well as at lower loads. The results of these measurements are plotted on the graph Fig. 4. The rectifier output volts plot yields an average supply impedance of 0.56 ohms. In the short circuit mode, 1 times R LIMIT, the 100 amp circuit breaker held 96 amps for five minutes and successfully interrupted the current when manually operated.

The inverter input voltage V2 as a function of current demand is also plotted. The resistor power dissipation line plots the power dissipated in the limit resistor as a function of current demand. The last plot, inverter input power, plots the power available at the inverter input terminal as a function of demanded current. The last two limit lines shall be discussed under the commutation tests.

The result of this first test proved that:

A) The power supply as configured would supply the required power, and

B) permit the normal protective firing mechanism to operate without causing disruption or damage to either the power supply or the test components.

9.3.3 Commutator Test

The objectives of this test were to:

A) measure the commutation losses and

b) to map the operational area of the inverter power supply combination.

In this test, the system was configured as in Fig. 3. The traction motor was replaced by a load resistor with three 2.37 ohm legs connected in a star configuration. The use of a resistive load rather than the motor load permitted rapid mapping of the frequency and power spectrum without the added complication and time required to monitor the motor acceleration and stabilization times. The first test was to measure the power required for inverter commutation with the inverter output voltage set to zero. The results of this test are plotted in Figures 5 and 6. Figure 5 plots the commutation input power against primary wave or output frequency. Figure 6 plots the power per switch cycle against the primary frequency. Additionally, the actual switching frequency is plotted against the primary output frequency.

The next test was to determine the commutation limits for the inverter. In a self-commutated inverter, such as the 13SG07, the limit on commutation is a function of both input voltage and output power. In order to determine the operational limits with the power suply configuration, a series of tests were run with the resistive load in Fig. 3 at three frequencies (5, 30 and 60 Hz). The output power was gradually increased until a protective firing was initiated due to a commutation failure. The results showed the upper working limit to be at 15.5 kW input and a V2 of 535 volts.

In order to avoid nuisance tripping problems, an operating limit of 5 kW and a V2 of 650 volts was defined. These two points are illustrated on the graph Fig. 4.

9.3.4 Motor Test

The final test was the operation of the traction motor. The objectives of this test were:

A) to determine the power level required by the motor and

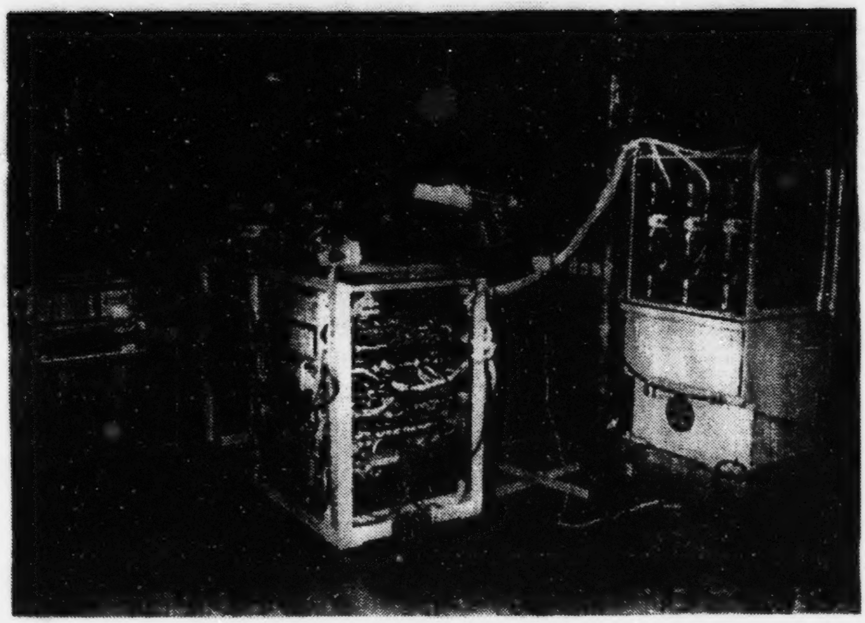
B) to determine if there are problems in an open loop mode.

For this test, the system was configured as in Fig. 2 with the traction motor connected directly to the inverter. A stroloscopic tachometer was used to monitor motor speed. In order to measure the power required, the motor was run up to maximum frequency speed and the voltage adjusted to maintain minimum synchronous lock. The results of this test are plotted in Fig. 7. As the graph shows, the maximum power draw was 880 watts. With the maximum commutation power draw of 8900 watts plus the motor draw, there is a power reserve of 5220 watts or a 5.9 to 1 power motion available for dynamic testing.

The second measurement was to determine if in open loop mode there would be any oscillation or instability in the control. The system was monitored both on the electrical and mechanical side. On the electrical side, no instabilities or oscillations were detected. On the mechanical side, a 0.5 Hertz first order rocking motion was excited at 30 rpm. Inspection of the motor mount showed that there was clearance at the case edges. Applying shims eliminated this oscillation and no additional modes were detected throughout the speed range.

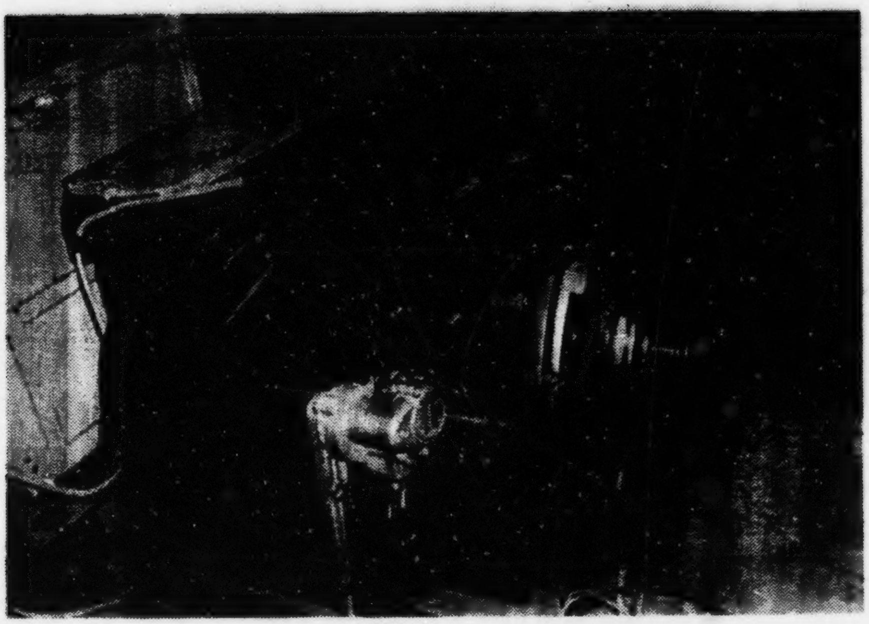
MICROPROCESSOR PROJECT

RIG COMMISSIONING



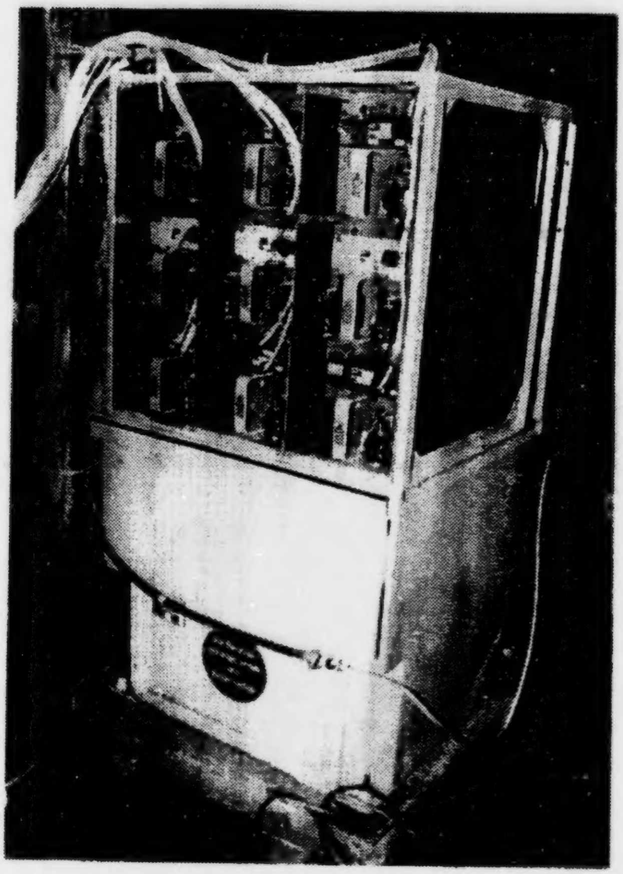
GENERAL VIEW

Fig. 9.1a



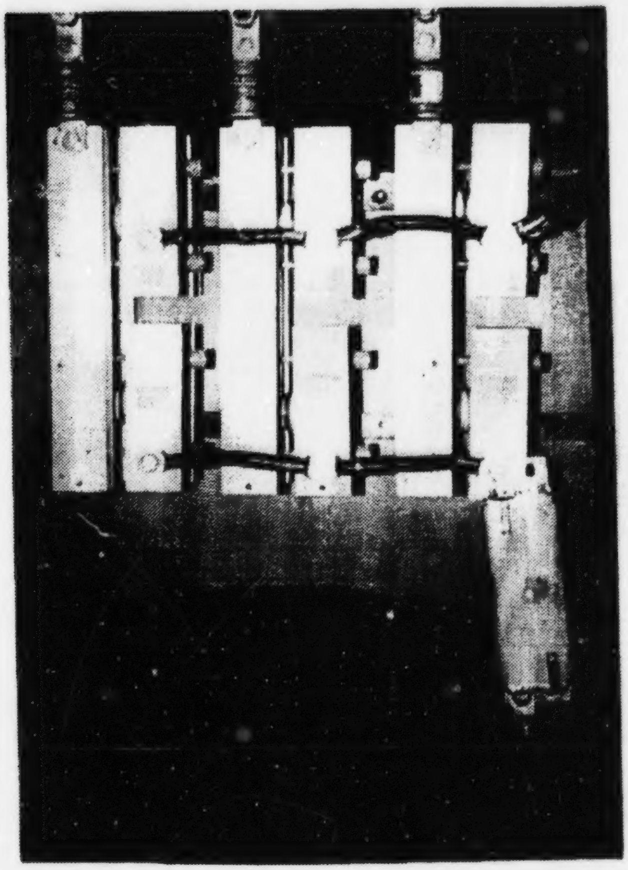
TRACTION MOTOR

Fig. 9.1b



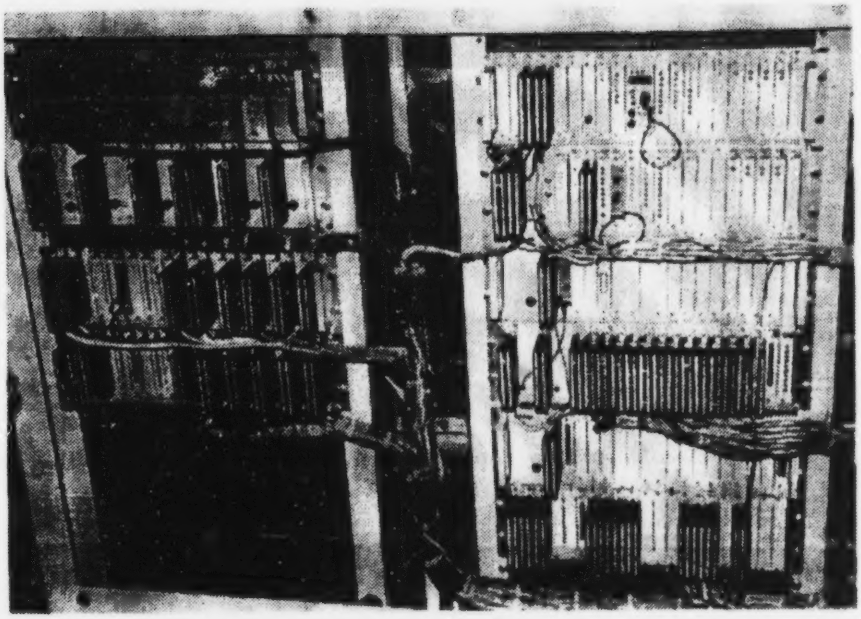
INVERTER

Fig. 9.1c



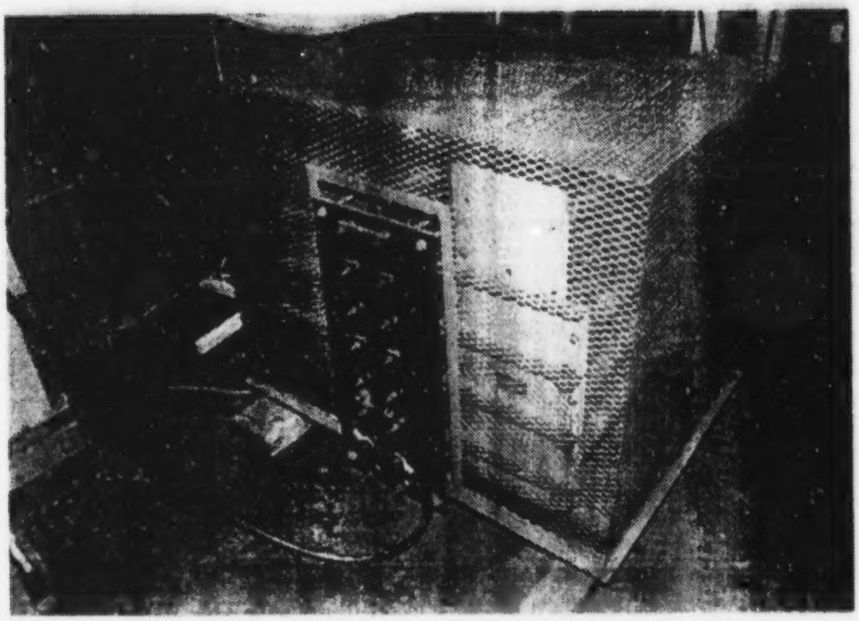
RECTIFIER

Fig. 9.1d



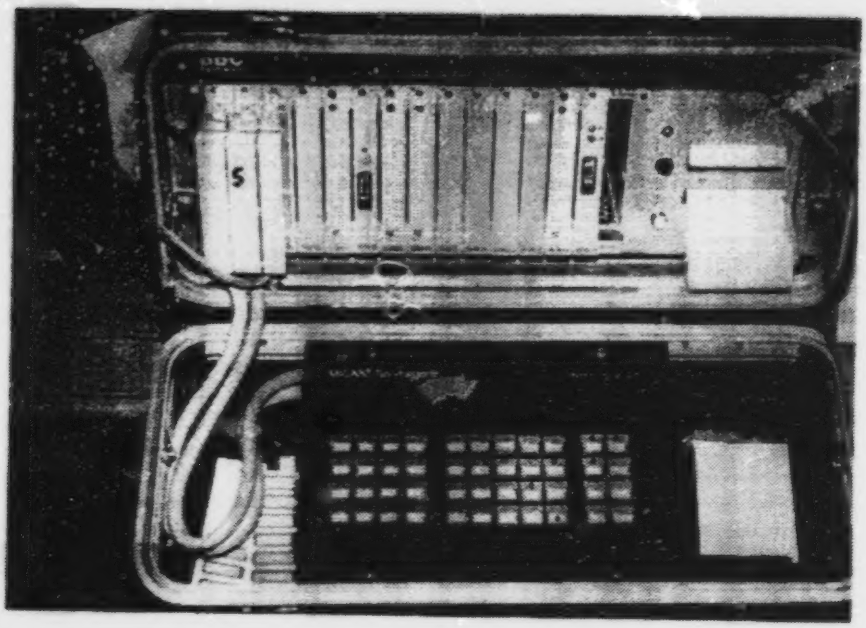
CONTROL

Fig. 9.1e



LIMIT RESISTOR

Fig. 9.1f



MICAS SERVICE DEVICE

Fig. 9.1g

MICROPROCESSOR TEST RIG

ELECTRICAL CONFIGURATION

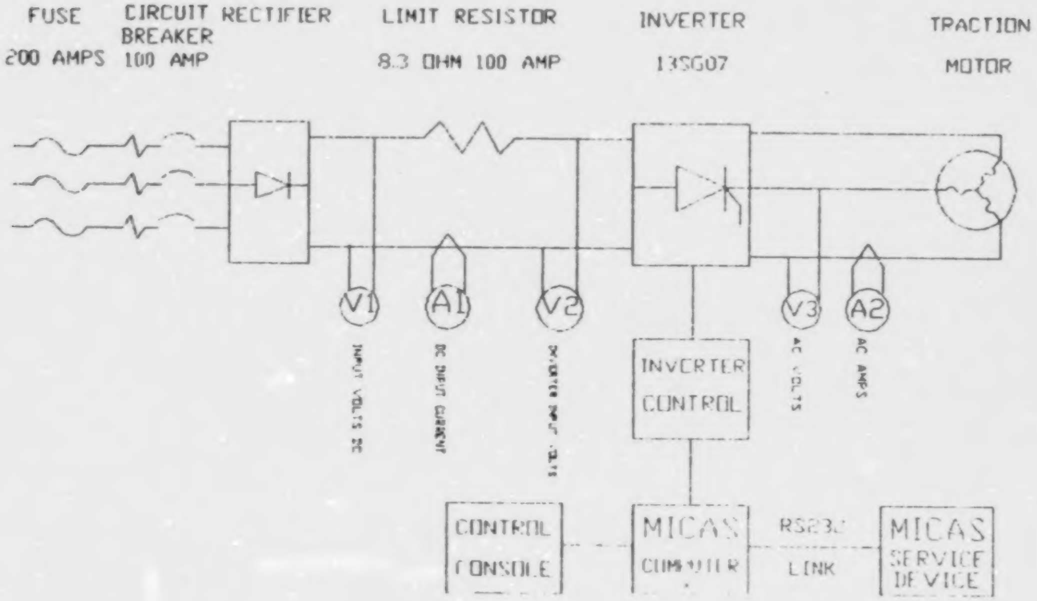

FUSE
200 AMPS
CIRCUIT BREAKER
100 AMP
RECTIFIER
LIMIT RESISTOR
8.3 OHM 100 AMP
INVERTER
13SG07
TRACTION MOTOR
V1
A1
V2
V3
A2
INPUT VOLTS DC
DC INPUT CURRENT
INVERTER INPUT VOLTS
AC VOLTS
AC AMPS
INVERTER CONTROL
CONTROL CONSOLE
MICAS COMPUTER
RS232 LINK
MICAS SERVICE DEVICE

Fig. 9.2

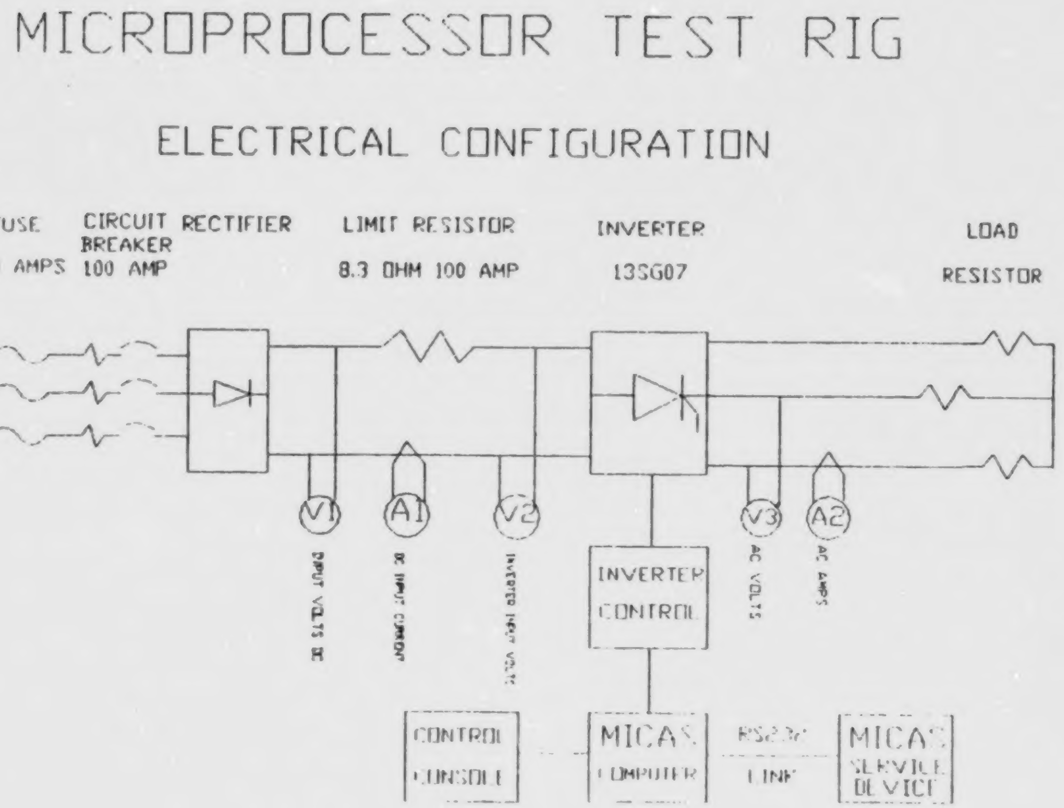

MICROPROCESSOR TEST RIG
ELECTRICAL CONFIGURATION
FUSE
200 AMPS
CIRCUIT BREAKER
100 AMP
RECTIFIER
LIMIT RESISTOR
8.3 OHM 100 AMP
INVERTER
13SG07
LOAD
RESISTOR
V1
A1
V2
V3
A2
INPUT VOLTS DC
DC INPUT CURRENT
INVERTER INPUT VOLTS
AC VOLTS
AC AMPS
INVERTER CONTROL
CONTROL CONSOLE
MICAS COMPUTER
RS232 LINK
MICAS SERVICE DEVICE

Fig. 9.3

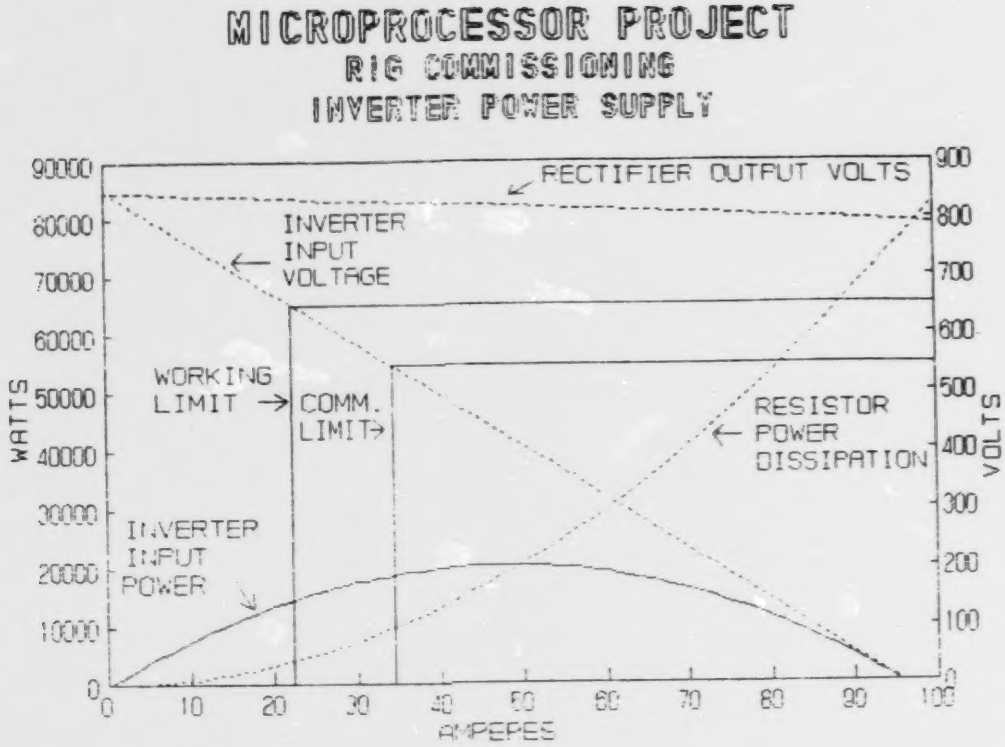

MICROPROCESSOR PROJECT
RIG COMMISSIONING
INVERTER POWER SUPPLY
RECTIFIER OUTPUT VOLTS
INVERTER INPUT VOLTAGE
WORKING LIMIT →
COMM. LIMIT→
RESISTOR POWER DISSIPATION
INVERTER INPUT POWER
WATTS
VOLTS
AMPERES
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
0
100
200
300
400
500
600
700
800
900
0
10
20
30
40
50
60
70
80
90
100

Fig. 9.4

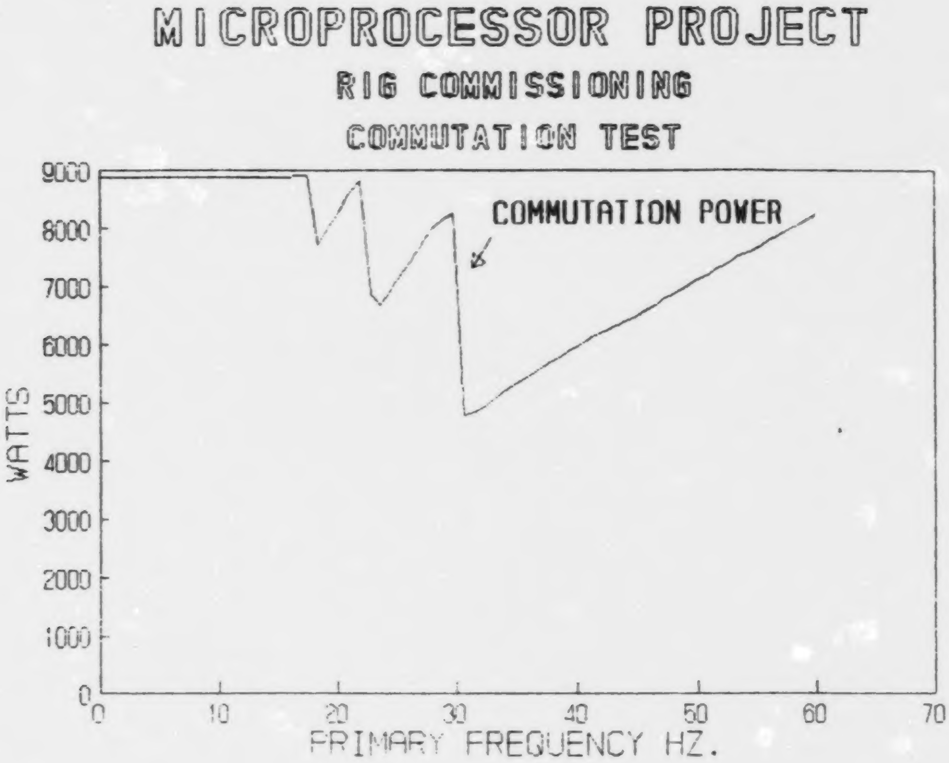

MICROPROCESSOR PROJECT
RIG COMMISSIONING
COMMUTATION TEST
COMMUTATION POWER
WATTS
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
0
10
20
30
40
50
60
70
PRIMARY FREQUENCY HZ.

Fig. 9.5

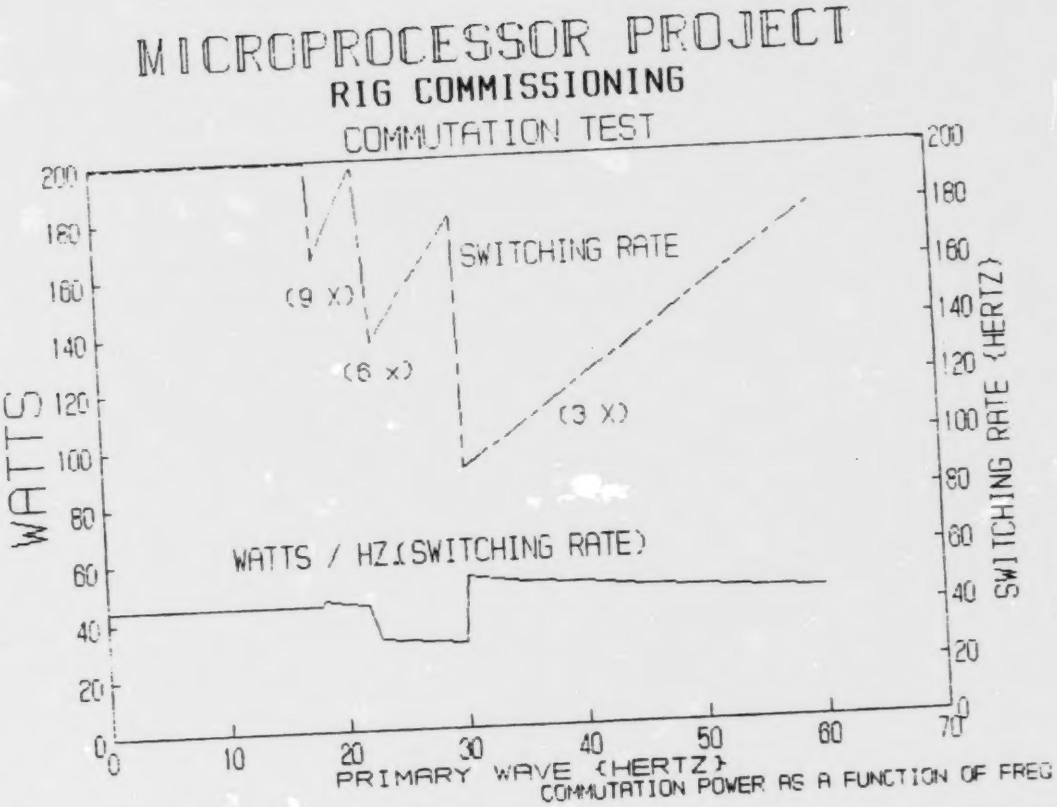

MICROPROCESSOR PROJECT
RIG COMMISSIONING
COMMUTATION TEST
WATTS
0
20
40
60
80
100
120
140
160
180
200
SWITCHING RATE
(9 X)
(6 x)
(3 X)
WATTS / HZ(SWITCHING RATE)
SWITCHING RATE {HERTZ}
0
20
40
60
80
100
120
140
160
180
200
0
10
20
30
40
50
60
70
PRIMARY WAVE {HERTZ}
COMMUTATION POWER AS A FUNCTION OF FREQ

Fig. 9.6

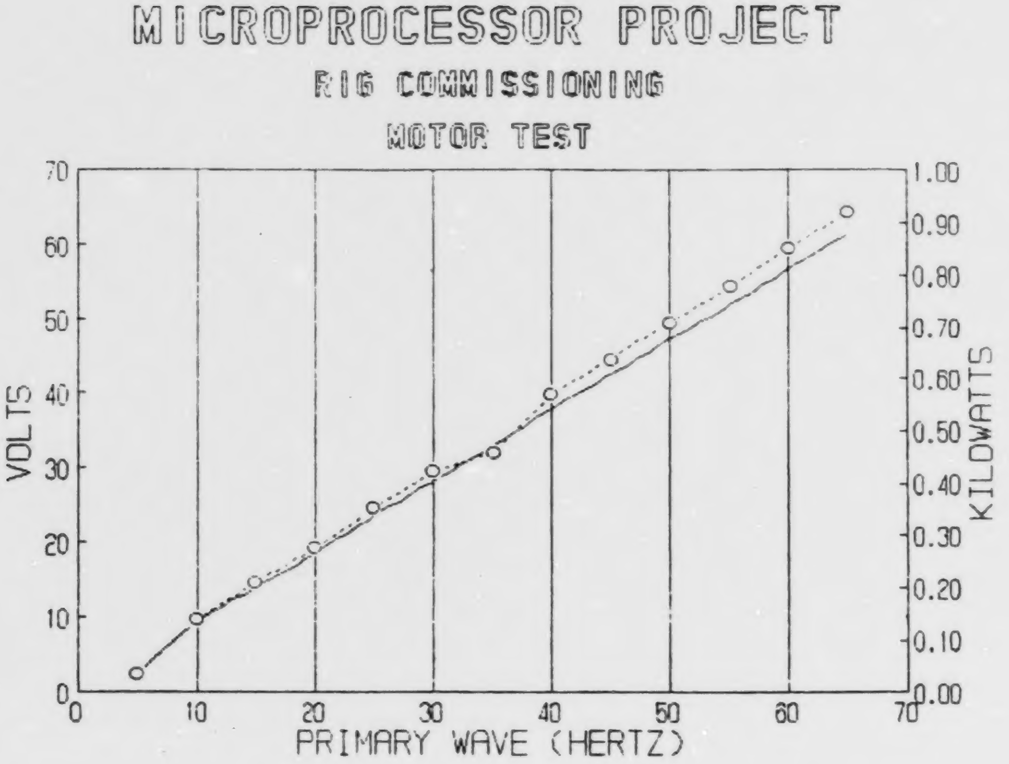

MICROPROCESSOR PROJECT
RIG COMMISSIONING
MOTOR TEST
VOLTS
KILOWATTS
PRIMARY WAVE (HERTZ)
0
10
20
30
40
50
60
70
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00

Fig. 9.7

10. Conclusions

Using microprocessors, it is possible to develop powerful support systems for the verification of control concepts and the testing of microprocessor based controllers. Such a system, the Real time Interactive Simulation and Test system (RIST) has been proposed and implemented. The system is based on a powerful 16/32 Bit microprocessor system with a wide variety of digital and analog interfaces and the possibility to be programmed in a high level programming language.

The RIST system was used to simulate the plant to be controlled. It was used as a realtime system connected to the MICAS microprocessor to verify and debug the developed software for the MICAS control system. It can be used with its real human interface, the locomotive console, for operator training and for the preparation of the commissioning. Its powerful interactive display and handling system provides efficient tools to the design and test engineer.

This concept is of general use to similar problems and other microprocessor control systems.

The RIST system was used to study several control problems and to test the proposed concepts and developed software for the microslip control, a data acquisition system and a diesel engine overload protection control.

The proposed microslip control takes advantage of the special characteristics of the ac drive system of the ac-dc-ac locomotive. The proposed control can handle a wide variety of slip conditions always controlling the microslip to the desired value. A maximum search option allows operation at increased microslip at maximum friction. The control does not rely on well defined friction characteristics and has no problems to operate with changing train inertia. This microslip control with the ac-dc-ac locomotive has the highest theoretical potential to utilize the maximum availble friction in locomotives with common bus feeding of the electric motors. The approach however requires that further advances be made in two other areas. First, the development of a non-contact ground speed pickup with an accuracy of better than 1%. The second requirement is a modification to the power conversion group to permit the rapid power modulations in response to the adhesion changes.

The proposed control for diesel engine overload protection protects the diesel in the case of limited power capacity due to internal faults. It can also provide protection against transient overload, which may occur if operation at increased microslip is desired. The control offers beyond the protection the possibility to utilize all availalbe diesel power, if this is not used by the auxiliaries. This system shows excellent promise, if it were to be interfaced to an electronic governing system for the prime mover.

To use the MICAS microprocessor control system for data acquisition and monitoring, a concept is proposed to combine the strength of the microprocessor control system with the data processing capabilities of a general computer. Using a Personal Computer with a general serial link provides a cost effective very flexible system. The use of the PC allows to utilize the wide variety of general software and interfaces for data processing, recording and display. The use of the serial link, with its low capacity, is possible through a concept, which uses data reduction at the source, by differing between continuous data and fault data. The system allows continuous monitoring as well as fault data analysis with high resolution in time. The necessary software for the MICAS System to accomplish this system has been developed and tested. From the work performed, it is apparent however that the current structure of the MICAS computer is not well suited to data storage. The information developed as a result of this program will be extremely useful in the design of a next generation computer system.

A study was carried out to determine the possibility of fuel savings by controls. There is only a limited potential for fuel savings by controls for multiple traction for locomotive operation with speed control.

It is suggested to do further investigations into train control systems. This holds probably the highest potential to improve fuel efficiency, and promises an increase of the all over performance of the transport system. Such an expert system for automatic train control could be part of the planned advance train control system to improve the North American railway system.

The possibility of testing the inverter control software using a traction inverter and traction motor, yet in a low power laboratory environment, was also demonstrated. It is capable of operating in the normal operating mode, including protection in complete safety. Additionally, the ability to operate the system in an open loop mode, a feature essential to control development, has also been successfully demonstrated.

References

(1) P. Katz, Digital Control Using Microprocessors, Prentice-Hall International, 1981.

(2) C. F. Logston, Jr., G. S. Itami, "Locomotive Friction-Creep Studies", ASME Paper No. 80-RT-1, April 1980.

(3) H. Fiehn, M. Weinhardt, N. Zeevenhooven, Drehstromversuchsfahrzeug der Niederlaendischen Eisenbahnen - Adhaesionsmessungen, Elektrische Bahnen 77, 1979.

(4) B. R. Meyer, A. P. de Buhr, "Super Series Wheel Creep Control System", ASME Paper No. 80-WA/RT-3, June 1980.

(5) Drehstromantriebstechnik fuer Schienenfahrzeuge, H. Strecker, W. Teich, BBC Druckschrift DVK701770 Sonderdruck aus Elsners Taschenbuch der Eisenbahntechnik 1975,76,77.

(6) J. Armstrong, KCS and the black box: A success (fuel saver equipment), Railway Age, April 1982.

(7) B. W. Parkinson, S. W. Gilbert, NAVSTAR: Global Positioning System - Ten Years Later, Proc. of the IEEE, Vol. 71, No. 10, October 1983.

(8) Train Dynamics Analyser, Freight Master^R Operator's Manual.

Appendix A1

Keyboard Commands for Simulation Control

Display Computer Keyboard:

C	clear display, no channels displayed
1,2,3,...8	toggles the display of channel 1,2,... on/off channels with data from the VME computer
M prefix followed by 1,2,3,4	displays on channel 1,2,... data from the MICAS instead of data from the VME
R	toggles: display and freeze display (run or stop 'chart recorder')
I	increment k ($k \geq 1$)
D	decrement k
	k determines the sample rate for the display and by this the time scale of the display.
	If the display is set (usually) to 250msec/division for $k = 1$ then pressing I 3 times results in $k+1+1+1=4$ and in $4 \cdot 250$msec = 1sec/division for the display.

List of Keyboard Controls to Control the Simulation

VME Computer Keyboard:

key	function
1,2,3,...9	set the transition time for TRGTI and TRGTO of the traction force characteristic to k * 200 msec k → key
P prefixed followed by 1,2,3,...9	sets train integrator (inertia) to k * 10sec k → key
<shift> 1,2,3	selects traction force characteristic 1, 2 or 3
F	toggles: load factor feedback on/off
L	toggles: diesel loaded by train or load off
S	starts transition from an old TRGTI and TRGTO value to the most recently set
O	toggles: maximum friction search on/off
A,B,C,D,E,I M,T,V,Z	selects certain sets of up to 7 variables to be transferred to the display from the VME computer

Appendix A2

Procedure Description and Listings of Programs
of Realtime Locomotive Simulation

A2.1	LOCO MODULE MAP, LOCO	
A2.2	DLOCO	(see typed)
A2.3	ELOCO	
A2.4	MLOCO	
A2.5	TLOCO	
A2.6	IOLOCO	
A2.7	ASMCURVE	
A2.8	USER_LIB	
A2.9	SYSM1	
A2.10	SYSM2	

LOCO MODULE MAP

MLOCO (Pascal)
< Motor Drive
Simulation executed
In PTMIRQ6 >
procedures:
MAX SEARCH
MOTORS:
- ELDRDELAY
- ACDRIVE
- MTRPLANT
- MLOSSES
- RAMPCHAR
- FCC

TLOCO (Pascal)
< Train Simulation >

procedures:
TRAIN:
- WINDAGE
- TRAINPLANT
- DGT, DGIB

ASMCURVE (Assembler)
< Profile Table >

Data Label:
- CURVESET

SYSTEM1 (Assembler)
< System Functions >
Subroutines:
- PUMULT
- APUDIV
- LPUMULT

SYSM2 (Assembler)
< Interrupt Vectors >
- INTSET

IOLOCO (Pascal)
< IO Module >
All memory mapped
I/O assignments

procedures:
SIGAH, SIGAL
SIGBH, SIGBL
IOSETUP
IO-PUTCHAR
IO-PUTSTRING
KEYBDINT
XMITINT
AC1AIRQ2
IRQ3
IODATA:
- DISPLINES
- DEBOUNCE
- ADINP
- DAOUT

LOCO (Pascal)
< Main Program Module >
All Global variable
declarations except
for memory mapped I/O

FUNCTIONS: PUDIV
ADJ
PROCEDURES:
CLOCK
RAMPGEN, DGM
PTMIRQ6
INITIALIZE
SYNCHRONIZE
PTMSETUP

DLOCO (Pascal)
< Diesel Simulation >

procedures:
DIESEL:
- DNOTCHAR
- DNDTINTGR
- TSD, TGEN
- TURBOFIX
- TSWITCH
- LFINTGR
- PLANT
- CHARAC
- GOVERNOR
- TFIX
- SHUTDOWN
- NODED1SUM
- DGA, DGB
- DGC, DGIA

ELOCO (Pascal)
< Electrical Simulation >

procedures:
ELECDRIVE:
- PECHAR
- PEINTGR
- VCHAR
- TGEN
- PNTCHAR
- TORQINV
- DGD, DGE
- DGI, DGT

USER-LIB (Assembler)
Monitor Routines
- Zend program
- Zstart program

PROCEDURE DESCRIPTIONS

Program LOCO

FUNCTION PUDIV (DVND, DVJR : PUINT) : PUINT;

Performs 16 bit per unit adjusted division as follows: PUDIV = DVND/DVSR checks are made for DVSR = 0, etc.

FUNCTION ADJ (SXTN : PUINT) : EIGHT;

Converters the signal sixteen bit per unit number SXTN to unsigned 8-bit binary offset representation for the purpose of data display.

Procedure CLOCK;

Generates a 1 Hz square wave for display on channel 8. ClkOutHigh and ClkOutLow determine the amplitude and offset of the square wave.

Procedure RAMPGEN (var OUT : PUINT) ;

(not presently active; see PJMJKQ6)

RampOn true which initiates a -1.0 to 1.0 PU ramp at OUT. When executed in a 1ms cycle, one complete ramp takes approximately 2.0 sec.

Procedure DGM;

Assignment of 'MOTORS' parameters to display channel array 'DATASET' are made. Conversion is made using the function ADJ. This procedure is executed in the 25ms cycle loop of the main program.

Procedure PTMIRQ6;

Interrupt routine for timer 1 with highest priority (level 6). Timer 1 is set to provide a 1ms continuous output. PTMIRQ6 establishes the base for the main program 25ms cycle time. Procedure MOTORS is executed in the interrupt routine to provide a fast 1ms cycle time.

Procedure INITIALIZE;

Numerous variables are initialized. Most are global: being initialized in the main module, but being used in another module. Some, like JOIN for example are global control parameters. Others, like OUTD1, are used in recursive difference equations (to implement an integrator, delay, etc.) and thus need a starting value. (usually zero) The cyclical nature of the simulation necessitates a separate initialization process *before* the actual cycle begins.

Procedure SYNCHRONIZE;

This procedure simply waits until the flag SLOWTIME is true (asserted by PTMIRQ6). This signals the beginning of a new 25ms cycle period and execution then proceeds.

Procedure PTMSETUP;

(Refer to pages 4-8, 4-9 of VME110 users guide). Timers 2,3 are disabled. Timer 1 is set for 1ms continuous output and started.

```
00000000  1      "68000"
00000000  1      $EXTENSIONS$
00000000  1
00000000  1  PROGRAM LOCO;
00000000  1
00000000  1
00000000  1  {   ****** LOCOMOTIVE SIMULATION  Ver 4.0 **** RALPH KNOX *****
00000000  1      ***** Dec 1985 *******
00000000  1
00000000  1      This program is a simulation of a diesel locomotive.  It
00000000  1      consists of ten modules, six of which are functionally
00000000  1      associated with a particular part of the simulation.
00000000  1      Thus there are six functional blocks: (1) the main module
00000000  1      (LOCO) which contains the main program, all global variable
00000000  1      definitions excepting the I/O variables, and all variable
00000000  1      initialization.  (2) the I/O module (IOLOCO) which contains
00000000  1      all routines that input or output data, condition and debounce
00000000  1      the data, or output messages and documentation to the monitor.
00000000  1      (3) the diesel module (DLOCO) which simulates a locomotive
00000000  1      diesel with speed controller and turbocharger.  (4) the torque
00000000  1      command control module (ELOCO) which conditions and controls
00000000  1      the torque and power requests. (5) the motor drive module
00000000  1      (MLOCO) which simulates the inverter, motor, and traction
00000000  1      force aspects of the locomotive and (6) the train module (TLOCO)
00000000  1      which simulates the the mass of the locomotive and factors
00000000  1      that affect the locomotive's movement.
00000000  1
00000000  1      The other four modules consist of ASMCURVE which contains the
00000000  1      traction force characteristics generated by an external program
00000000  1      CURVEGEN, USER_LIB which is an assembly language module that
00000000  1      handles various system functions on the VME 68000 unit, and
00000000  1      SYSM1 and SYSM2 which are also assembly language modules
00000000  1      containing vector table and interrupt set up routines, as well
00000000  1      as per unit multiplication and division routines.
00000000  1
00000000  1      All data in the program is based on a fixed point, 2's
00000000  1      complement per unit representation and has the data type
00000000  1      definition PUINT.  1.0  per unit  is equal to 16383
00000000  1      (2**14 - 1).  Variables of this form function as INTEGERS
00000000  1      except during multiplication and division where special
00000000  1      assembler routines have been implemented.
00000000  1
00000000  1      A real-time visual display system has been incorporated
00000000  1      into the simulation.  Various data sets consisting of
00000000  1      eight 8-bit variables can be sent through an interrupting
00000000  1      communication channel to an external 6809-based system
00000000  1      which displays the eight channels on an oscilloscope.
00000000  1
00000000  1                                                                   }
00000000  1
```

```
 51 00000000  1  CONST   TSH = 03H;      { = 1 ms sample rate = 804  }
 52 00000000  1          TSL = 20H;
 53 00000000  1          N = 8;          { = no. of channels being monitored }
 54 00000000  1          M = 100;        { = no. of samples/ channel  }
 55 00000000  1          L = 800;        { = total no. of samples / sweep  }
 56 00000000  1
 57 00000000  1          R0 = 1;         { write Ctrl reg.s 1,3 }
 58 00000000  1          R1 = 3;         { read status, write Ctrl reg. 2 }
 59 00000000  1          R2 = 5;         { read T1C, write MSB buffer }

 60 00000000  1          R3 = 7;         { read LSB buffer, write T1L }
 61 00000000  1          R4 = 9;         { read T2C, write MSB buffer }
 62 00000000  1          R5 = 11;        { read LSB buffer, write T2L }
 63 00000000  1          R6 = 13;        { read T3C, write MSB buffer }
 64 00000000  1          R7 = 15;        { read LSB buffer, write T3L }
 65 00000000  1
 66 00000000  1          T1BIT = 0;
 67 00000000  1          ENINTS = 2000H;
 68 00000000  1
 69 00000000  1  TYPE    PUINT = SIGNED_16;     { signed 16 bit per unit repr. }
 70 00000000  1          INTEGER = SIGNED_16;
 71 00000000  1          BYTE = SIGNED_8;
 72 00000000  1          EIGHT = UNSIGNED_8;
 73 00000000  1          LONGINT = SIGNED_32;
 74 00000000  1          LPUINT = SIGNED_32;
 75 00000000  1          BITS = SET OF 0..7;
 76 00000000  1          MEMIO = ARRAY [0..16] OF BYTE;
****WARNING ??                                   ^508
 77 00000000  1
 78 00000000  1  VAR     SLOWCOUNT : INTEGER;
 79 00000002  1          SLOWTIME : BOOLEAN;
 80 00000003  1
 81 00000003  1          CLKCOUNT : INTEGER;
 82 00000006  1          HIGH : BOOLEAN;
 83 00000007  1
 84 00000007  1          PSET,NSET : PUINT;
 85 0000000C  1          TDSET,PESET : PUINT;
 86 00000010  1          PTMTR,EXTST : PUINT;
 87 00000014  1          RAMP : PUINT;
 88 00000016  1
 89 00000016  1          NDIESEL,PDIESEL : PUINT;
 90 0000001A  1          LDFCTR : PUINT;
 91 0000001C  1          PDLIM : PUINT;
 92 0000001E  1
 93 0000001E  1          TEREF : PUINT;
 94 00000020  1          NMOT,TMOT : PUINT;
 95 00000024  1          NLOCOSET : PUINT;
 96 00000026  1
 97 00000026  1          FLOCO,PLOAD : PUINT;
 98 0000002A  1          NLOCO : PUINT;
 99 0000002C  1
```

```
100 0000002C  1            TRGTI,TRGTO : PUINT;
101 00000030  1            SLOP : PUINT;
102 00000032  1
103 00000032  1
104 00000032  1            K : INTEGER;
105 00000034  1            LOOPK : INTEGER;
106 00000036  1            II : INTEGER;
107 00000038  1
108 00000038  1           $GLOBVAR ON$
109 00000038  1
110 00000038  1            IP,OP : INTEGER;
111 0000003C  1            SELECT : CHAR;
112 0000003D  1            JOIN : BOOLEAN;
113 0000003E  1            KFCTR : INTEGER;
114 00000040  1            RAMPSTART : BOOLEAN;
115 00000041  1            PINDEX : INTEGER;
116 00000044  1            RAMPON : BOOLEAN;
117 00000045  1            LINEFD : BOOLEAN;

118 00000046  1            LFIN : BOOLEAN;
119 00000047  1            EXTFCTR : INTEGER;
120 0000004A  1
121 0000004A  1            DUMREAD : BYTE;
122 0000004B  1            I : INTEGER;
123 0000004E  1            DATASET : ARRAY [1..N] OF EIGHT;
124 00000056  1
125 00000056  1            MESSAGE : STRING;
126 00000156  1
127 00000156  1            MP1,MP2 : EIGHT;
128 00000158  1            SM1,SM2 : EIGHT;
129 0000015A  1            PSETCNT,SWCNT : INTEGER;
130 0000015E  1            CDBINPA,CDBINPB : EIGHT;
131 00000160  1
132 00000160  1
133 00000160  1            OUTN1,INPN1 : PUINT;
134 00000164  1            NODED2 : PUINT;
135 00000166  1
136 00000166  1            OUTT1,INPT1 : PUINT;
137 0000016A  1            TURBSET,TDELAY : PUINT;
138 0000016E  1            UCNT : INTEGER;
139 00000170  1
140 00000170  1            OUTLF1,INPLF1 : PUINT;
141 00000174  1            LF1 : PUINT;
142 00000176  1
143 00000176  1            OUTD1,INPD1,NODED1 : PUINT;
144 0000017C  1            STRTFLG,NTCH1FLG : BOOLEAN;
145 0000017E  1            STRTHYS,DLED : BOOLEAN;
146 00000180  1
147 00000180  1
148 00000180  1            OUTPE1,INPPE1 : PUINT;
149 00000184  1            PEINIT : PUINT;
```

```
150 00000186  1          OUTW1,INPW1 : PUINT;
151 0000018A  1          WINIT : PUINT;
152 0000018C  1          MOTORING,BRAKING : BOOLEAN;
153 0000018E  1          INTSLT,EXTSLT : BOOLEAN;
154 00000190  1          TROOP : BOOLEAN;
155 00000191  1          OUTSC1,INPSC1 : PUINT;
156 00000196  1          MSCIN : BOOLEAN;
157 00000197  1          LECTIME : INTEGER;
158 0000019A  1
159 0000019A  1          OUTE1,INPE1 : PUINT;
160 0000019E  1          OUTLD1,INPLD1 : PUINT;
161 000001A2  1          INPM1 : PUINT;
162 000001A4  1          OUTM1 : LPUINT;
163 000001A8  1          NMREF,F1 : PUINT;
164 000001AC  1          TEREFD : PUINT;
165 000001AE  1          F2,NMLIM : PUINT;
166 000001B2  1          SLIP,LMOT : PUINT;
167 000001B6  1
168 000001B6  1          RAMPRUN : BOOLEAN;
169 000001B7  1          MPRI,MPRO : PUINT;
170 000001BC  1          LOCALTIME : INTEGER;
171 000001BE  1          NSEA : BOOLEAN;
172 000001BF  1          TMOTOLD : PUINT;
173 000001C2  1
174 000001C2  1          NLREF : PUINT;
175 000001C4  1          OUTL1,INPL1 : PUINT;
176 000001C8  1          LOCOTIME : INTEGER;

177 000001CA  1
178 000001CA  1
179 000001CA  1          $ORG 00006BH$
180 000001CA  1           VECTOR2 : UNSIGNED_32;  { Autovector 2 in INT. table }
181 000001CA  1           VECTOR3 : UNSIGNED_32;  {     "      3 "   "     "  }
182 000001CA  1          $ORG 00007BH$
183 000001CA  1           VECTOR6 : UNSIGNED_32;  {     "      6 "   "     "  }
184 000001CA  1
185 000001CA  1
186 000001CA  1          $ORG 0FE8010H$       { memory map-- timer module }
187 000001CA  1           PTM : MEMIO;
188 000001CA  1
189 000001CA  1          $ORG 0FE8013H$
190 000001CA  1           PTMSTATUS : BITS;  { set version }
191 000001CA  1
192 000001CA  1          $GLOBVAR OFF$
193 000001CA  1
194 000001CA  1          $END_ORG$
195 00000000  1
196 00000000  1
197 00000000  1
198 00000000  1  FUNCTION PUMULT(XER,XAND : PUINT): PUINT; EXTERNAL;
199 00000000  1
```

```
200 00000000  1      { PUMULT performs 16 bit, per unit adjusted
201 00000000  1        multiplication.
202 00000000  1        Assembly file listing :
203 00000000  1
204 00000000  1                 PROG
205 00000000  1                 GLOBAL PUMULT           ; function PUMULT
206 00000000  1                 GLOBAL APUDIV
207 00000000  1       PUMULT   LINK      A6,#-2
208 00000000  1                MOVE.W    10[A6],D0
209 00000000  1                MULS      8[A6],D0       ; get product
210 00000000  1                ASL.L     #2,D0          ; adjust PU result
211 00000000  1                SWAP      D0             ; get upper word
212 00000000  1                MOVE.W    D0,-2[A6]
213 00000000  1                MOVE.W    -2[A6],D7
214 00000000  1                UNLK      A6
215 00000000  1       RPUMULT  RTS                                   }
216 00000000  1
217 00000000  1
218 00000000  1
219 00000000  1  FUNCTION LPUMULT(XER,XAND : PUINT): LPUINT; EXTERNAL;
220 00000000  1
221 00000000  1      { LPUMULT performs 16 bit, per unit adjusted
222 00000000  1        multiplication but retains the lower 16 LSB's
223 00000000  1        and thus passes the product as a 32 bit value.
224 00000000  1      Assembly file listing:
225 00000000  1
226 00000000  1                NAME "MULTS. 2 16 BIT PU NUMBERS, PRODUCT = 32 BIT"
227 00000000  1       LPUMULT  LINK      A6,#-4
228 00000000  1                MOVE.W    8[A6],D0
229 00000000  1                MULS      10[A6],D0
230 00000000  1                ASL.L     #2,D0
231 00000000  1                MOVE.L    D0,-4[A6]
232 00000000  1                MOVE.L    -4[A6],D7
233 00000000  1                UNLK      A6
234 00000000  1       RLPUMULT RTS                                    }
235 00000000  1

236 00000000  1
237 00000000  1  PROCEDURE SIGAH; EXTERNAL;
238 00000000  1
239 00000000  1  PROCEDURE SIGAL; EXTERNAL;
240 00000000  1
241 00000000  1  {   PROCEDURE SIGBH; EXTERNAL;
242 00000000  1
243 00000000  1  PROCEDURE SIGBL; EXTERNAL;   }
244 00000000  1
245 00000000  1  $GLOBPROC ON$
246 00000000  1
247 00000000  1  FUNCTION PUDIV(DVND,DVSR : PUINT): PUINT;
248 00000000  2
249 00000000  2      { PUDIV performs 16 bit, per unit adjusted
250 00000000  2        division.  Since quotient overflow is possible,
251 00000000  2        no division is performed if DIVIDEND > DIVISOR   }
252 00000000  2
253 00000000  2  CONST    MAXPU = 16383;  { 1.0 PU  }
254 00000000  2           Delta0 = 1;     { a very small value }
255 00000002  2
256 00000002  2  VAR      QUOTIENT : INTEGER;
257 00000000  2
258 00000000  2
259 00000000  2
260 00000000  2  FUNCTION APUDIV(DVND,DVSR : PUINT); PUINT; EXTERNAL;
261 00000004  2
262 00000004  2     { Assembly file listing :
263 00000004  2
264 00000004  2     APUDIV   LINK      A6,#-2
265 00000004  2              CLR.L     D0
266 00000004  2              MOVE.W    10[A6],D0
267 00000004  2              SWAP      D0
268 00000004  2              ASR.W     #2,D0
269 00000004  2              DIVS      8[A6],D0
270 00000004  2              MOVE.W    D0,-2[A6]
271 00000004  2              MOVE.W    -2[A6],D7
272 00000004  2              UNLK      A6
273 00000004  2     RAPUDIV  RTS                          }
274 00000004  2
275 00000004  2  BEGIN
276 00000004  2     IF ( ABS(DVND) < ABS(DVSR) ) AND (DVSR <> 0) THEN
277 00000030  2          BEGIN
278 00000030  2               QUOTIENT := APUDIV(DVND,DVSR); { divide as usual }
279 00000042  2
280 00000042  2               PUDIV := QUOTIENT;   { assign funtion value}
281 00000048  2
282 00000048  2               IF (QUOTIENT = 0) AND (DVND <> 0) THEN
283 00000062  2                    REPEAT UNTIL K <> 0; {STOP!,answer incorrect}
284 0000006B  2          END
285 0000006C  2     ELSE
286 0000006C  2          IF DVND = 0 THEN
287 00000074  2               PUDIV := 0
288 0000007C  2          ELSE
289 0000007C  2               BEGIN
290 0000007C  2                    IF DVSR = 0 THEN
291 00000084  2                         DVSR := Delta0; { give dvsr small value }
292 0000008A  2                    IF PUMULT(DVND,DVSR) < 0 THEN
293 0000009E  2                         PUDIV := - MAXPU    { -ve limit    }
294 000000A8  2                    ELSE

295 000000A8  2                         PUDIV := MAXPU;     { +ve limit    }
296 000000AE  2               END;
297 000000AE  2  END; { PUDIV }
298 00000000  1
299 00000000  1
```

```
300 00000000  1
301 00000000  1  FUNCTION ADJ(SXTN : PUINT): EIGHT;
302 00000000  2
303 00000000  2      { ADJ converts a signed sixteen bit PU number to an
304 00000000  2        unsigned 8 bit number in positive offset
305 00000000  2        representation for purposes of data display.    }
306 00000000  2
307 00000000  2  CONST    CNVRT = UNSIGNED_8(80H);
308 00000000  2           SHIFTB = UNSIGNED_16(128);
309 00000001  2
310 00000001  2  VAR      UNS : EIGHT;
311 000000BC  2
312 000000BC  2  BEGIN
313 000000BC  2      UNS := UNSIGNED_16(SXTN) / SHIFTB;
314 000000CA  2      IF UNS >= CNVRT THEN
315 000000D2  2           ADJ := UNS - CNVRT
316 000000DE  2      ELSE
317 000000DE  2           ADJ := UNS + CNVRT;
318 000000EA  2  END; { ADJ }
319 00000000  1
320 00000000  1  $GLOBPROC OFF$
321 00000000  1
322 00000000  1
323 00000000  1
324 00000000  1  PROCEDURE IO_PUTCHAR(Ch : CHAR); EXTERNAL;
325 00000000  1
326 00000000  1
327 00000000  1
328 00000000  1  PROCEDURE IO_PUTSTRING(VAR OutString : STRING); EXTERNAL;
329 00000000  1
330 00000000  1
331 00000000  1
332 00000000  1       {   Due to high gain in the feedback path of the motor
333 00000000  1           area of the simulation, the integrator MTRPLANT
334 00000000  1           needs to be implemented with a fast cycle time.     }
335 00000000  1
336 00000000  1
337 00000000  1
338 00000000  1  PROCEDURE CLOCK;
339 00000000  2
340 00000000  2      {  Generates a 1 Hz square wave displayed on
341 00000000  2         channel 8 so that a time base is established.  }
342 00000000  2
343 00000000  2  CONST    HalfCycle = 20;     { = 500 Ts = 0.5 sec. }
344 00000000  2           ClkOutHigh = -9830; { -0.6 PU    }
345 00000000  2           ClkOutLow = -13107;  {-0.8 PU    }
346 00000000  2
347 00000000  2  VAR      Clk : PUINT;
348 000000F8  2
349 000000F8  2  BEGIN
350 000000F8  2
```

```
351 000000F8  2      CLKCOUNT := CLKCOUNT + 1;
352 000000FC  2
353 000000FC  2      IF CLKCOUNT = HalfCycle THEN

354 00000108  2           BEGIN
355 00000108  2                CLKCOUNT := 0;
356 0000010C  2                HIGH := NOT HIGH;
357 00000112  2                IF HIGH THEN
358 0000011A  2                     Clk := ClkOutHigh
359 00000124  2                ELSE
360 00000124  2                     Clk := ClkOutLow;
361 0000012A  2
362 0000012A  2                DATASET [8] := ADJ(Clk);
363 00000136  2           END;
364 00000136  2
365 00000136  2  END; ( CLOCK )
366 00000000  1
367 00000000  1
368 00000000  1
369 00000000  1
370 00000000  1
371 00000000  1  (     interrupt routines for:
372 00000000  1            1) keyboard interrupts for "data group control"
373 00000000  1            2) transmit buffer interrupts for outputting
374 00000000  1               messages to the terminal.
375 00000000  1
376 00000000  1  PROCEDURE ACIAIRQ2; EXTERNAL;
377 00000000  1
378 00000000  1
379 00000000  1
380 00000000  1      ( Interrupt procedure outputs member of DS   )
381 00000000  1
382 00000000  1  PROCEDURE IRQ3; EXTERNAL;
383 00000000  1
384 00000000  1
385 00000000  1
386 00000000  1  PROCEDURE INTSET(STTSWRD:INTEGER); EXTERNAL;
387 00000000  1
388 00000000  1  (     INTSET sets up the interrupt mask in the
389 00000000  1        SYSTEM BYTE of the STATUS REGISTER.
390 00000000  1        Assembly file listing:
391 00000000  1
392 00000000  1
393 00000000  1                 ."68000"
394 00000000  1                 PROG
395 00000000  1                 NAME "INTERRUPT AND VECTOR SET UP"
396 00000000  1        EIRQ1    EQU       $-1
397 00000000  1                 GLOBAL    INTSET
398 00000000  1                 EXTERNAL  PTMIRQ6,VECTOR6
399 00000000  1                 EXTERNAL  IRQ3,VECTOR3
400 00000000  1                 EXTERNAL  ACIAIRQ2,VECTOR2
```

```
401 00000000  1        INTSET   LINK      A6,#-2
402 00000000  1                 MOVE.L    #PTMIRQ6,VECTOR6    ; load autovector6
403 00000000  1                 MOVE.L    #IRQ3,VECTOR3       ; load autovector3
404 00000000  1                 MOVE.L    #ACIAIRQ2,VECTOR2   ; load autovector2
405 00000000  1                 MOVE.W    8(A6),SR       ; set interrupt level
406 00000000  1                 UNLK      A6
407 00000000  1                 RTS
408 00000000  1                                                      }
409 00000000  1
410 00000000  1
411 00000000  1  PROCEDURE RAMPGEN(VAR OUT: PUINT);
412 00000000  2

413 00000000  2  CONST    TimeInc = 08;
414 00000000  2           IncLim = 16383;
415 0000013E  2
416 0000013E  2  BEGIN
417 0000013E  2     IF RAMPON THEN
418 00000146  2         BEGIN
419 00000146  2             OUT := OUT + TimeInc;
420 00000150  2             IF OUT >= IncLim THEN
421 0000015E  2                 BEGIN
422 0000015E  2                     OUT := -16383;
423 00000166  2                     RAMPON := FALSE;
424 0000016A  2                 END;
425 0000016A  2         END;
426 0000016A  2  END; { RAMPGEN }
427 00000000  1
428 00000000  1
429 00000000  1  {     ****** FUNCTIONAL BLOCK PROCEDURES ******
430 00000000  1     Each to the following procedures perform a specific
431 00000000  1     function in the locomotive simulation.  Functional
432 00000000  1     blocks have been separated into separate modules.      }
433 00000000  1
434 00000000  1
435 00000000  1  PROCEDURE IODATA(VAR PSET,NSET,TDSET,PESET,PTMTR,PDLIM: PUINT;
436 00000000  2                VAR TRGTI,TRGTO,EXTST,NLOCOSET : PUINT;
437 00000000  2                NMOT,NDIESEL,LDFCTR,NLOCO : PUINT); EXTERNAL;
438 00000000  1
439 00000000  1
440 00000000  1
441 00000000  1  PROCEDURE DIESEL(NSET,TDSET,PLOAD,PTMTR : PUINT;
442 00000000  2                VAR NDIESEL,PDIESEL,LDFCTR : PUINT); EXTERNAL;
443 00000000  1
444 00000000  1
445 00000000  1
446 00000000  1  PROCEDURE ELECDRIVE(    TEST,PESET,NDIESEL,PSET,NMOT,
447 00000000  2                         TMOT,LDFCTR,NLOCO,NLOCOSET : PUINT;
448 00000000  2                   VAR TEREF,DLOAD : PUINT); EXTERNAL;
449 00000000  1
450 00000000  1
```

```
00000000  1
00000000  1
00000000  1
00000000  1  (   Simulates the traction induction motor drives
00000000  1      Note that MOTORS is executed in the 1 ms interrupt
00000000  1      procedure PTMIRQ6 and not in the cycle loop of the
00000000  1      main program.                                  )
00000000  1
00000000  1  PROCEDURE MOTORS(    TEST,TEREF,NLOCO,TRGTI,TRGTO,SLOP : PUINT;
00000000  2                       PROFILENUM : INTEGER;
00000000  2                   VAR TMOT,NMOT,FLOCO : PUINT); EXTERNAL;
00000000  1
00000000  1
00000000  1
00000000  1  (   MaxSearch is functionally part of the motor drive simulation.
00000000  1      It is not however executed at the 1ms cycle time that MOTORS
00000000  1      is.  For this reason, MaxSearch is located in NLOCO but
00000000  1      is called in the 25ms main program loop.                 )
00000000  1
00000000  1  PROCEDURE MAXSEARCH(NMOT,NLOCO,TMOT,FSPOT : PUINT;
00000000  2                       VAR SLOP : PUINT); EXTERNAL;

00000000  1
00000000  1
00000000  1
00000000  1  (   DGM is executed in the cycle loop of the main program
00000000  1      for timing reasons.                                 )
00000000  1
00000000  1  PROCEDURE DGM;
00000172  2
00000172  2  BEGIN
00000172  2      DATASET [1] := ADJ(TMOT);
00000180  2      DATASET [2] := ADJ(F1);     ( F1 <=> SLIP )
0000018E  2      DATASET [3] := ADJ(PTMTR); ( PTMTR <=> NMREF )
0000019C  2      DATASET [4] := ADJ(NLOCO);
000001AA  2      DATASET [5] := ADJ(F2);   ( F2 <=> FLOCO )
000001B8  2      DATASET [6] := ADJ(NMOT);
000001C6  2      DATASET [7] := ADJ(NMLIM);    ( NMLIM <=>  LMOT )
000001D4  2  END; (DGM)
00000000  1
00000000  1
00000000  1
00000000  1
00000000  1
00000000  1      (    This procedure simulates the "train".  This includes
00000000  1           the locomotives's inertia and wind resistance. )
00000000  1
00000000  1  PROCEDURE TRAIN(TEST,FLOCO : PUINT; VAR NLOCO : PUINT); EXTERNAL;
000001CA  1
000001CA  1
000001CA  1
```

```
501 000001CA  1
502 000001CA  1  $GLOBPROC ON$
503 000001CA  1  $INTERRUPT ON$
504 00000000  1
505 00000000  1  PROCEDURE PTMIRQ6;
506 00000000  2
507 00000000  2  (   This interrupt routine provides a "fast cycle"
508 00000000  2      time base using a 1 ms interrupt output from
509 00000000  2      timer 1.   The procedure MOTORS, which requires
510 00000000  2      high speed sampling is executed during this
511 00000000  2      interrupt procedure.                        )
512 00000000  2
513 00000000  2  CONST    SlowCyc = 25;
514 000001E0  2
515 000001E0  2  BEGIN
516 000001E0  2
517 000001E0  2      REPEAT (  ensure timer1 source of interrupt )
518 000001E0  2      UNTIL T1BIT IN PTMSTATUS;
519 000001EA  2
520 000001EA  2      DUMREAD := PTM [R2];  ( clear T1 interrupt )
521 000001F2  2
522 000001F2  2  (   RAMPGEN(RAMP);    )
523 000001F2  2
524 000001F2  2      MOTORS(PTMTR,TEREF,NLOCO,TRGTI,TRGTO,SLOP,PINDEX,
525 000001F2  2          (  ^^^^^  )                     TMOT,NMOT,FLOCO);
526 00000224  2          (  RAMP   )
527 00000224  2      SLOWCOUNT := SLOWCOUNT + 1;
528 00000228  2      IF SLOWCOUNT = SlowCyc THEN
529 00000234  2          BEGIN
530 00000234  2              SLOWCOUNT := 0;

531 00000238  2              SLOWTIME := TRUE;
532 0000023E  2          END;
533 0000023E  2
534 0000023E  2  END; ( PTMIRQ6 )
535 00000000  1
536 00000000  1  $INTERRUPT OFF$
537 00000000  1  $GLOBPROC OFF$
538 00000000  1
539 00000000  1
540 00000000  1
541 00000000  1  PROCEDURE INITIALIZE;
542 00000000  2
543 00000000  2      (    Global variables  are initialized.  Most
544 00000000  2           globals would otherwise be local to some
545 00000000  2           procedure save for the necessity of initializing
546 00000000  2           at the beginning.                            )
547 00000000  2
548 00000000  2  CONST    US0 = EIGHT(0);
549 0000024A  2
550 0000024A  2  BEGIN
```

```
0000024A  2      II := 0;
0000024E  2      I := 1;          ( initialize "pointer" to DATASET )
00000254  2
00000254  2      SELECT := 'A';  ( default data group A )
0000025A  2
0000025A  2      JOIN := TRUE;   ( default load condition )
00000260  2      RAMP := -16384; ( init. for test ramp in )
00000266  2      RAMPON := FALSE; ( MLOCO                 )
0000026A  2
0000026A  2      SLOWCOUNT := 0;       ( 25 ms cycle counter )
0000026E  2      SLOWTIME := FALSE;    ( 25 ms start indicator )
00000272  2
00000272  2      CLKCOUNT := 0;  ( initialize "clock" parameters )
00000276  2      HIGH := FALSE;
0000027A  2
0000027A  2      IP := 1;         ( input position index in output queue )
00000280  2      OP := 1;         ( output   "      "   "    "     "  )
00000286  2      LINEFD := FALSE;     ( LF, CR control flag )
0000028A  2
0000028A  2      MP1 := US0;        ( initialize PSETCODE debouncing storage )
0000028E  2      MP2 := US0;
00000292  2      SM1 := US0;        ( initialize SWCODE debouncing storage )
00000296  2      SM2 := US0;
0000029A  2
0000029A  2      PSETCNT := 0;      ( initialize PSET debounce counter )
0000029E  2      SWCNT := 0;        ( initialize SW debounce counter )
000002A2  2      CDBINPA := US0;     ( initialize debounced PSET input )
000002A6  2      CDBINPB := US0;     ( initialize debounced SW input )
000002AA  2
000002AA  2      OUTD1 := 0;     ( initialize diesel plant storage values )
000002AE  2      INPD1 := 0;
000002B2  2      NODED1 := 0;    ( initialize diesel plant input )
000002B6  2
000002B6  2      OUTN1 := 0;     ( initialize dn/dt limiter integrator values)
000002BA  2      INPN1 := 0;
000002BE  2      NODED2 := 0;    ( initialize dn/dt intgr input  )
000002C2  2
000002C2  2      OUTT1 := 0;     ( initialize turbo delay storage values  )
000002C6  2      INPT1 := 0;

000002CA  2      TURBSET := 0;   ( initialize turbo delay input )
000002CE  2      TDELAY := 0;
000002D2  2      UCNT := 1;      ( initialize slow sample counters )
000002D8  2
000002D8  2      OUTLF1 := 0;    ( initialize load factor intgr values )
000002DC  2      INPLF1 := 0;
000002E0  2      LF1 := 0;       ( initialize load factor intgr input )
000002E4  2
000002E4  2      OUTE1 := 0;     ( init. elec. drive delay storage values )
000002E8  2      INPE1 := 0;
000002EC  2      OUTLD1 := 0;    ( init. loco feedback delay storage values )
```

```
601 000002F0  2      INPLD1 := 0;
602 000002F4  2      NMOT := 0;       { init. motor speed }
603 000002F8  2
604 000002F8  2      PEINIT := 0;     { init. torque limiter input value }
605 000002FC  2      OUTPE1 := 0;     { init. torque limiter intgr storage values }
606 00000300  2      INPPE1 := 0;
607 00000304  2
608 00000304  2      WINIT := 0;      { init. W limiter input value }
609 00000308  2      OUTW1 := 0;      { init. W limiter intgr storage values }
610 0000030C  2      INPW1 := 0;
611 00000310  2
612 00000310  2      LFIN := TRUE;    { load factor switched in at ELOCO }
613 00000316  2      TEREF := 0;
614 0000031A  2      OUTM1 := 0;      { init. motor plant storage values }
615 0000031E  2      INPM1 := 0;
616 00000322  2      PLOAD := 0;       { init. actual load power  }
617 00000326  2
618 00000326  2      OUTSC1 := 0;     { init. speed control integrator }
619 0000032A  2      INPSC1 := 0;
620 0000032E  2      LSCTIME := 0;    { init. speed control scale factor }
621 00000332  2
622 00000332  2      KFCTR := 1;            { init. for multiplier ramps }
623 00000338  2      RAMPSTART := FALSE;  { in MLOCO                    }
624 0000033C  2      RAMPRUN := FALSE;
625 00000340  2      MPRI := 16383;
626 00000346  2      MPRO := 16383;
627 0000034C  2      LOCALTIME := 0; { MAXSEARCH relative cycle time counter}
628 00000350  2      NSEA := TRUE;    { init. new search flag }
629 00000356  2      MSCIN := FALSE;  { MaxSearch OUT }
630 0000035A  2
631 0000035A  2      FLOCO := 0;      { init. locomotive friction force }
632 0000035E  2      NMREF := 0;
633 00000362  2
634 00000362  2      NLOCO := 0;      { init. ground speed value. }
635 00000366  2      OUTL1 := 0;      { init. train plant storage values }
636 0000036A  2      INPL1 := 0;
637 0000036E  2      NLREF := 0;      {  "      "      "   input }
638 00000372  2      LOCOTIME := 0;  { init. relative train Ts counter  }
639 00000376  2      EXTFCTR := 6;   { train plant set to 6 * 10s }
640 0000037C  2
641 0000037C  2      K := 0;
642 00000380  2
643 00000380  2      PINDEX := 1;     { default profile selection }
644 00000386  2
645 00000386  2
646 00000386  2  END; { INITIALIZE }
647 00000000  1
648 00000000  1

649 00000000  1
650 00000000  1  PROCEDURE SYNCHRONIZE;
```

```
0000038E  2
0000038E  2      (    Execution waits for end of timer-- indicating
0000038E  2           beginning of next sample period.  Present
0000038E  2           cycle time is 25 ms.
0000038E  2                                                          }
0000038E  2
0000038E  2  BEGIN
0000038E  2
0000038E  2           REPEAT         ( wait for sample time )
0000038E  2           UNTIL SLOWTIME;
00000394  2
00000394  2           SLOWTIME := FALSE;
00000398  2
00000398  2  END; ( SYNCHRONIZE )
00000000  1
00000000  1
00000000  1
00000000  1  PROCEDURE IOSETUP; EXTERNAL;
00000000  1
00000000  1
00000000  1
00000000  1  PROCEDURE PTMSETUP;
000003A0  2
000003A0  2  BEGIN
000003A0  2
000003A0  2  (    set up TIMER1 for sample cycle     )
000003A0  2
000003A0  2      PTM [R1] := 00H;     ( access to CR3 )
000003A6  2      PTM [R0] := 00H;     ( disable T3    )
000003AC  2      PTM [R1] := 01H;     ( disable T2    )
000003B4  2
000003B4  2      PTM [R0] := 0C2H;    ( set up TIMER1 )
000003BC  2      PTM [R2] := TSH;     ( load T1 with Ts )
000003C4  2      PTM [R3] := TSL;     ( and start timer )
000003CC  2
000003CC  2  END; ( PTMSETUP )
000003DC  1
000003DC  1
000003DC  1
000003DC  1
000003DC  1  BEGIN ( main program )
000003DC  1
000003DC  1      IOSETUP;     ( set up IO ports )
000003E0  1
000003E0  1      PTMSETUP;    ( set up interrupt timers )
000003E2  1
000003E2  1      INITIALIZE;          ( set inputs and storage values to 0 )
000003E6  1
000003E6  1      INTSET(ENINTS);      ( set up CPU ints )
000003F0  1
000003F0  1
```

```
 702 000003F0  1      MESSAGE := '$$            LOCOMOTIVE SIMULATION VER 4.0$$';
 703 000003FC  1      IO_PUTSTRING(MESSAGE);
 704 00000406  1
 705 00000406  1
 706 00000406  1
 707 00000406  1      LOOPK := 0;

 708 0000040A  1
 709 0000040A  1
 710 0000040A  1      { Simulation algorithm begins ...  }
 711 0000040A  1
 712 0000040A  1      WHILE LOOPK = 0 DO  { repeat indefinitely }
 713 00000412  1      BEGIN
 714 00000412  1
 715 00000412  1           SIGAL;
 716 00000416  1
 717 00000416  1           SYNCHRONIZE;
 718 0000041A  1
 719 0000041A  1           SIGAH;
 720 0000041E  1
 721 0000041E  1           CLOCK;
 722 00000422  1
 723 00000422  1
 724 00000422  1           IODATA(PSET,NSET,TDSET,PESET,PTMTR,PDLIM,TRGTI,TRGTO,
 725 00000422  1                  EXTST,NLOCOSET,NMOT,NDIESEL,LDFCTR,NLOCO);
 726 00000464  1
 727 00000464  1           DIESEL(NSET,TDSET,PLOAD,EXTST,NDIESEL,PDIESEL,LDFCTR);
 728 0000048A  1
 729 0000048A  1
 730 0000048A  1           ELECDRIVE(PTMTR,PESET,NDIESEL,PSET,NMOT,TMOT,LDFCTR,
 731 0000048A  1                                     NLOCO,NLOCOSET,TEREF,PLOAD);
 732 000004C0  1      {  Note: MOTORS is shown here for completeness only. }
 733 000004C0  1
 734 000004C0  1      { MOTORS(TEST,TEREF,NLOCO,TRGTI,TRGTO,SLOP,PINDEX,
 735 000004C0  1                                     TMOT,NMOT,FLOCO);    }
 736 000004C0  1
 737 000004C0  1
 738 000004C0  1           MAXSEARCH(NMOT,NLOCO,TMOT,PTMTR,SLOP);
 739 000004DE  1
 740 000004DE  1           IF SELECT = 'M' THEN DGM;
 741 000004EE  1
 742 000004EE  1
 743 000004EE  1           TRAIN(EXTST,FLOCO,NLOCO);
 744 00000500  1
 745 00000500  1      END;
 746 00000504  1
 747 00000504  1  END. { main program }
 748 00000000  1
 749 00000000  1

508: Warning: field or entry alignment; record or array comparisons may not work
```

```
End of compilation, number of errors=    0
```

Program DLOCO

Procedure DNDTCHAR (INP : PUINT ; var OUT : PUINT)

Proportional controller with P = 50 from -1.0 to 1.0 PU. Together with DNDTINTER, a d/dt limiter is created to restrict the maximum allowable rate of change of the incoming speed request signal.

Procedure DNDTINTGR (INPN φ : PUINT ; VAR OUTN φ : PUINT)

Performs the integrator function $H(s) = 1/5s$. The integrator winds up to 1.0 PU or down to -1.0 PU. (see DNDTCHAR)

Procedure TSD (INPT φ : PUINT ; var OUTT φ : PUINT)

Part of the Turbocharger sub-block section. This block performs a first order delay $H(s) = \frac{1}{1 + T_S}$ where T is equal to 1.0 sec for ascending inputs and 0.5s for descending inputs.

Procedure TURBOFIX (INP, TDSET : PUINT ; var OUT : PUINT)

Provides torque limiting characteristic for the turbocharge circuit. The output torque value is limited by TDSETA (the requested value) or 0.314, whichever is higher.

Procedure TGEN (ACTRQ, SPEED : PUINT ; VAR TURBOTCR : PUINT)

Part of the turbocharge circuit, an available torque value is created as a function of the diesel speed and the current shaft torque value.

Procedure TSWITCH (INP, T1, T2 : PUINT ; VAR OUT : PUINT)

Part of the load factor signal generation circuit. When T2 > T1, the speed control error signal - 0.03 PU is fed to an integrator circuit.

Procedure LFCHAR (INP : PUINT ; VAR OUT : PUINT)

Characteristic shaping signal from TSWITCH before going to the integrator. IF the INP > 0 then OUT = INP. IF $INP \geq 0$, OUT = 0.2 INP.

Procedure LFINTGR (INPLF φ : PUINT ; VAR OUTLF φ : PUINT)

Performs the integrator function $H(s) = 1/0.1s$. The output is limited between 0 and 1 PU.

Procedure PLANT (INPD φ : PUINT ; VAR OUTD φ : PUINT)

Integrator models the diesel engine by performing the function $H(s) = 1/1s$.

Procedure CHARAC (INP : PUINT ; VAR OUT : PUINT)

Characteristic for diesel. If the integrator is below 0.3 PU, the output is set to 0.0. Integrator output is limited to 1.0 PU.

Procedure DLOSSES (INP : PUINT ; VAR OUT : PUINT)

Losses in the diesel are calculated as a function of diesel speed.

Procedure GOVERNOR (INP : PUINT ; VAR OUT : PUINT)

Proportional Controller with P = 30 works in the speed control loop.

Procedure TFIX (INP : PUINT ; VAR OUT : PUINT)

The shaft torque is limited by a torque limit value which is the lesser of the requested torque (TDSET) or the turbocharger torque value (TURBSETD).

Procedure SHUTDOWN (INP : PUINT ; VAR OUT : PUINT)

Enforces that diesel can only be started in notch 1. Torque is added to speed start up and a negative torque is added to stall the diesel when speed drops below 0.3 PU.

Procedure NODED1SUM (IN1, IN2, IN3 : PUNIT ; VAR OUT : PUINT)

The particular node has three signals being summed. To avoid overflow, summing is done double precision and then truncated.

Procedure DGA; data set procedures
DGB assign various internal
DGC values to real time
DGIA display system.

```
  1 00000000  1  "68000"
  2 00000000  1  $EXTENSIONS$
  3 00000000  1
  4 00000000  1  PROGRAM DLOCO;
  5 00000000  1
  6 00000000  1
  7 00000000  1      {    DLOCO is an external module containing the procedure
  8 00000000  1           DIESEL which in turn contains all the procedures
  9 00000000  1           for the diesel portion of the locomotive simulation.
 10 00000000  1                 The EXTERNAL variables listed below represent
 11 00000000  1           globals from the main program module which are used
 12 00000000  1           within this module.
 13 00000000  1                                                                     }
 14 00000000  1
 15 00000000  1  CONST    N = 8;
 16 00000000  1
 17 00000000  1  TYPE     PUINT = SIGNED_16;
 18 00000000  1           INTEGER = SIGNED_16;
 19 00000000  1           BYTE = SIGNED_8;
 20 00000000  1           EIGHT = UNSIGNED_8;
 21 00000000  1           LPUINT = SIGNED_32;
 22 00000000  1           LONGINT = SIGNED_32;
 23 00000000  1           STATUSBITS = SET OF 0..7;
 24 00000000  1           PROFILE = ARRAY [0..255] OF PUINT;
 25 00000000  1           TABLE = ARRAY [1..5] OF PROFILE;
 26 00000000  1
 27 00000000  1
 28 00000000  1  $EXTVAR ON$
 29 00000000  1
 30 00000000  1  VAR      OUTN1,INPN1 : PUINT;
 31 00000000  1           OUTT1,INPT1 : PUINT;
 32 00000000  1           UCNT : INTEGER;
 33 00000000  1           OUTLF1,INPLF1 : PUINT;
 34 00000000  1           OUTD1,INPD1 : PUINT;
 35 00000000  1           STRTHYS,NTCH1FLG : BOOLEAN;
 36 00000000  1           STRTFLG,DLED : BOOLEAN;
 37 00000000  1           NODED1,NODED2 : PUINT;
 38 00000000  1           TDELAY,TURBSET : PUINT;
 39 00000000  1           LF1 : PUINT;
 40 00000000  1           JOIN : BOOLEAN;
 41 00000000  1           SELECT : CHAR;
 42 00000000  1           DATASET : ARRAY [1..N] OF EIGHT;
 43 00000000  1
 44 00000000  1  $EXTVAR OFF$
 45 00000000  1
 46 00000000  1
 47 00000000  1
 48 00000000  1  FUNCTION PUMULT(XER,XAND : PUINT): PUINT; EXTERNAL;
 49 00000000  1
 50 00000000  1  FUNCTION PUDIV(DVND,DVSR : PUINT): PUINT; EXTERNAL;
 51 00000000  1
 52 00000000  1  FUNCTION ADJ(SITN : PUINT): EIGHT; EXTERNAL;
 53 00000000  1
 54 00000000  1
 55 00000000  1
 56 00000000  1  $GLOBPROC ON$
 57 00000000  1
 58 00000000  1
 59 00000000  1  PROCEDURE DIESEL(NSET,TDSET,PLOAD,TEST : PUINT;

 60 00000000  2                   VAR NDIESEL,PDIESEL,LDFCTR : PUINT);
 61 00000000  2
 62 00000000  2
 63 00000000  2  VAR      NODEDO,NSETLIM,NODED3 : PUINT;
 64 00000006  2           NERR,TABS,TREF : PUINT;
 65 0000000C  2           TSHAFT,TURBOTOR,LFIN : PUINT;
 66 00000012  2           DLOUT,SSOUT,TDLOAD : PUINT;
 67 00000000  2
 68 00000000  2
 69 00000000  2
 70 00000000  2  PROCEDURE DNDTCHAR(INP : PUINT; VAR OUT : PUINT);
 71 00000000  3
 72 00000000  3     {    Proportional controller with P = 50 provides
 73 00000000  3          an upper and lower integrating input value.
 74 00000000  3          This procedure along with DNDTINTGR limit the
 75 00000000  3          rate at which the speed setting NSET may change.   }
 76 00000000  3
 77 00000000  3  CONST    P = 50;
 78 00000000  3           MAX = 16383;
 79 00000000  3           MAX_50 = 327;      { MAX/50 = 0.020 PU  }
 80 00000004  3
 81 00000004  3  BEGIN
 82 00000004  3     IF INP > MAX_50 THEN
 83 00000010  3          OUT := MAX    { go to upper limit }
 84 0000001C  3     ELSE
 85 0000001C  3          IF INP >= (- MAX_50) THEN
 86 00000028  3              OUT := P * INP      { proportional region }
 87 00000036  3          ELSE
 88 00000036  3              OUT := - MAX;       { go to lower limit }
 89 0000003E  3  END; { DNDTCHAR }
 90 00000000  2
 91 00000000  2
 92 00000000  2
 93 00000000  2  PROCEDURE DNDTINTGR(INPNO : PUINT; VAR OUTNO : PUINT);
 94 00000000  3
 95 00000000  3     {    performs the integrator function:
 96 00000000  3                         OUTNO   =   1
 97 00000000  3                         -----     -----
 98 00000000  3                         INPNO      5 S         }
 99 00000000  3
100 00000000  3  CONST    A = 41;   { 0.0025 PU }
```

```
101 00000000  3          ABSMAX = 16383;     { 1.0 PU }
102 00000000  3          ABSMIN = -16383;    { - "" }
103 00000046  3
104 00000046  3  BEGIN
105 00000046  3
106 00000046  3     OUTNO := OUTN1 + PUMULT(A,INPNO) + PUMULT(A,INPN1);
107 00000078  3
108 00000078  3     IF OUTNO > ABSMAX THEN
109 00000086  3          OUTNO := ABSMAX;
110 0000008E  3     IF OUTNO < ABSMIN THEN
111 0000009C  3          OUTNO := ABSMIN;
112 000000A4  3
113 000000A4  3     OUTN1 := OUTNO;
114 000000AC  3     INPN1 := INPNO;
115 000000B2  3
116 000000B2  3  END; { DNDTINTGR }
117 00000000  2
118 00000000  2

119 00000000  2
120 00000000  2  PROCEDURE TSD(INPTO : PUINT; VAR OUTTO : PUINT);
121 00000000  3
122 00000000  3     {    gives the delay characteristics associated
123 00000000  3          with the torque generated by a turbocharger.
124 00000000  3          The delay function is:
125 00000000  3                         OUTTO   =     1
126 00000000  3                         -----       --------
127 00000000  3                         INPTO        1 + T S
128 00000000  3
129 00000000  3          where T = 1.0 sec for ascending inputs,
130 00000000  3          and T = 0.5 sec for descending inputs.         }
131 00000000  3
132 00000000  3
133 00000000  3  CONST    UFCTR = 2;  { rising delay = 2 * 0.5 s }
134 00000000  3           DFCTR = 1;   { falling delay = 1 * 0.5 s }
135 00000000  3           A = 15585;    { T = 0.5s  }
136 00000000  3           B = 400;
137 00000000  3           ABSMAX = 16383;
138 00000000  3           ABSMIN = 0;
139 000000BA  3
140 000000BA  3  BEGIN
141 000000BA  3     IF OUTT1 < INPTO THEN  {   rising delay   }
142 000000C6  3          IF UCNT = UFCTR THEN
143 000000D2  3          BEGIN
144 000000D2  3               OUTTO := PUMULT(A,OUTT1)
145 000000D2  3               + PUMULT(B,INPTO) + PUMULT(B,INPT1);
146 00000114  3               UCNT := 1;
147 0000011A  3
148 0000011A  3               IF OUTTO > ABSMAX THEN OUTTO := ABSMAX;
149 00000130  3               IF OUTTO < ABSMIN THEN OUTTO := ABSMIN;
150 00000140  3               OUTT1 := OUTTO;
```

```
0000014B  3                 INPT1 := INPT0;
0000014E  3             END
00000152  3             ELSE
00000152  3                 UCNT := UCNT + 1
0000015A  3        ELSE
0000015A  3             BEGIN
0000015A  3                 OUTT0 := PUMULT(A,OUTT1)
0000015A  3                 + PUMULT(B,INPT0) + PUMULT(B,INPT1);
0000019E  3                 IF OUTT0 > ABSMAX THEN OUTT0 := ABSMAX;
000001B4  3                 IF OUTT0 < ABSMIN THEN OUTT0 := ABSMIN;
000001C4  3                 OUTT1 := OUTT0;
000001CC  3                 INPT1 := INPT0;
000001D2  3             END
000001D2  3   END; { TSD }
00000000  2
00000000  2
00000000  2
00000000  2   PROCEDURE TURBOFIX(INP,TDSET : PUINT; VAR OUT : PUINT);
00000000  3
00000000  3        {      provides torque characteristic for
00000000  3               turbocharger.                          }
00000000  3
00000000  3
00000000  3   CONST    MIN = 5147;   { 0.3142 PU = Notch 4 }
00000000  3
00000000  3   VAR      UPLIM : PUINT;
000001DA  3

000001DA  3   BEGIN
000001DA  3       IF INP > MIN THEN         { input > notch 4 }
000001E6  3            BEGIN
000001E6  3                 IF TDSET > MIN THEN      { determine limit }
000001F2  3                     UPLIM := TDSET
000001FA  3                 ELSE
000001FA  3                     UPLIM := MIN;
00000200  3
00000200  3                 IF INP > UPLIM THEN      { limit input }
0000020C  3                      OUT := UPLIM
00000218  3                 ELSE
00000218  3                      OUT := INP;
00000220  3            END
00000224  3       ELSE                         { input < notch 4 }
00000224  3            OUT := MIN;
0000022C  3
0000022C  3   END; { TURBOFIX }
00000000  2
00000000  2
00000000  2
00000000  2   PROCEDURE TGEN(ACTRQ,SPEED : PUINT; VAR TURBOTOR : PUINT);
00000000  3
00000000  3        {      A torque is determined based on the speed
00000000  3            of the diesel.  This torque represents the
00000000  3            possible torque that can be delivered by the
00000000  3            turbocharger.                              }
00000000  3
00000000  3  CONST    MaxRated = 15639;   { 0.9546 PU = 1050 RPM }
00000000  3           M = 2403;       { 0.1467 PU = slope }
00000000  3           B = 17367;      { 1.06 PU = y int. }
00000000  3           ABSMAX = 16383;      { 2 **15 -1 }
00000000  3           MaxMltplr = 19661; { 1.2 PU }
00000000  3
00000000  3  VAR      MLTPLR : PUINT;
00000234  3
00000234  3  BEGIN
00000234  3
00000234  3     IF SPEED > MaxRated THEN
00000240  3         MLTPLR := MaxMltplr
0000024A  3     ELSE
0000024A  3         MLTPLR := PUMULT(M,SPEED) + B;
00000260  3
00000260  3     TURBOTOR :=  PUMULT(MLTPLR,ACTRQ);
00000274  3
00000274  3  END;
00000000  2
00000000  2
00000000  2
00000000  2  PROCEDURE TSWITCH(INP,T1,T2 : PUINT; VAR OUT: PUINT);
00000000  3
00000000  3     {     If T2 > T1 then the error speed (delta NDIESEL)
00000000  3           is subtracted from .03 PU to provide a load
00000000  3           factor control line.                       }
00000000  3
00000000  3  CONST    Ref = 492;      { 0.03 PU }
00000000  3
00000000  3  VAR      DeltaN : PUINT;
0000027C  3
0000027C  3  BEGIN

0000027C  3
0000027C  3     IF T1 > T2 THEN
00000288  3         DeltaN := INP
00000292  3     ELSE
00000292  3         DeltaN := 0;
00000296  3
00000296  3     OUT := Ref - DeltaN;
000002A4  3
000002A4  3  END; { TSWITCH }
00000000  2
00000000  2
00000000  2
00000000  2  PROCEDURE LFCHAR(INP : PUINT; VAR OUT : PUINT);
00000000  3
```

```
00000000  3  CONST    MP = 3277;      ( 0.2 PU )
00000000  3           MAX = 16383;    ( 1.0 PU )
00000000  3           MIN = -16383;   ( -1.0 PU )
000002AC  3
000002AC  3  BEGIN
000002AC  3      IF INP ) MAX THEN
000002B8  3           OUT := MAX            ( limit output to MAX )
000002C4  3      ELSE
000002C4  3           IF INP )= 0 THEN
000002CC  3                OUT := PUMULT(MP,INP)  ( positive linear range )
000002E4  3           ELSE
000002E4  3                IF INP ) MIN THEN
000002F0  3                     OUT := INP     ( negative linear range )
000002FA  3                ELSE
000002FA  3                     OUT := MIN;    ( limit output to MIN )
00000302  3  END; ( LFCHAR )
00000000  2
00000000  2
00000000  2
00000000  2  PROCEDURE LFINTGR(INPLFO : PUINT; VAR OUTLFO : PUINT);
00000000  3
00000000  3      (    performs the integrator function:
00000000  3                            OUTLFO    =     1
00000000  3                            ------      -------
00000000  3                            INPLFO       (0.1)S                 )
00000000  3
00000000  3  CONST    A = 2059;             ( 0.1256 PU )
00000000  3           ABSMAX = 16383;       ( 1.0 PU )
00000000  3           ABSMIN = 0;
0000030A  3
0000030A  3  BEGIN
0000030A  3
0000030A  3      OUTLFO := OUTLF1 + PUMULT(A,INPLFO) + PUMULT(A,INPLF1);
0000033C  3
0000033C  3      IF OUTLFO ) ABSMAX THEN
0000034A  3           OUTLFO := ABSMAX;
00000352  3      IF OUTLFO ( ABSMIN THEN
0000035C  3           OUTLFO := ABSMIN;
00000362  3
00000362  3      OUTLF1 := OUTLFO;
0000036A  3      INPLF1 := INPLFO;
00000370  3
00000370  3  END; ( LFINTGR )
00000000  2
00000000  2


00000000  2
00000000  2  PROCEDURE PLANT(INPDO : PUINT; VAR OUTDO : PUINT);
00000000  3
00000000  3      (  performs the integrator function:
00000000  3                           OUTDO    =     1
```

```
301 00000000  3                              ----        -----
302 00000000  3                              INPDO       (1)S            )
303 00000000  3
304 00000000  3  CONST    A = 205;           ( = 0.0125 PU )
305 00000000  3           ABSMAX = 16383;    ( = 1.0 PU )
306 00000000  3           ABSMIN = 0;        ( = 0.00 PU   )
307 00000378  3
308 00000378  3  BEGIN   ( INTEGRATE  )
309 00000378  3
310 00000378  3     OUTDO := OUTD1 + PUMULT(A,INPDO) + PUMULT(A,INPD1);
311 000003AA  3
312 000003AA  3     IF OUTDO > ABSMAX THEN  ( LIMIT OUTDO TO ABSMAX )
313 000003B8  3         OUTDO := ABSMAX;
314 000003C0  3     IF OUTDO < ABSMIN THEN
315 000003CA  3         OUTDO := ABSMIN;
316 000003D0  3
317 000003D0  3     OUTD1 := OUTDO;
318 000003D8  3     INPD1  := INPDO;
319 000003DE  3
320 000003DE  3  END; ( PLANT )
321 00000000  2
322 00000000  2
323 00000000  2
324 00000000  2  PROCEDURE CHARAC(INP : PUINT; VAR OUT : PUINT);
325 00000000  3
326 00000000  3     ( provides a limit characteristic for the
327 00000000  3       integrator output                              )
328 00000000  3
329 00000000  3  CONST    MINOP = 4915;       ( 0.3 PU )
330 00000000  3           MAX = 16383;        ( 1.0 PU )
331 00000000  3           MIN = 0;            ( 0.0 PU )
332 000003E6  3
333 000003E6  3  BEGIN
334 000003E6  3     IF INP < MINOP THEN        ( stall range )
335 000003F2  3         OUT := MIN
336 000003FC  3     ELSE
337 000003FC  3         BEGIN
338 000003FC  3             IF INP > MAX THEN     ( limit range )
339 00000408  3                 OUT := MAX
340 00000414  3             ELSE                  ( linear range )
341 00000414  3                 OUT := INP;
342 0000041C  3         END;
343 0000041C  3  END; ( CHARAC )
344 00000000  2
345 00000000  2
346 00000000  2
347 00000000  2  PROCEDURE DLOSSES(INP : PUINT; VAR OUT : PUINT);
348 00000000  3
349 00000000  3     ( Calculates simulated diesel losses from
350 00000000  3       integrator output; .03 PU < LOSSES < .05 PU      )
351 00000000  3
```

```
00000000  3  CONST    M = 328;  { SLOPE = .02 PU   }
00000000  3           B = 492;  { Y INTCP = .03 PU  }
00000424  3

00000424  3  BEGIN
00000424  3      OUT := PUMULT(M,INP) + B;
0000043C  3  END; { DLOSSES }
00000000  2
00000000  2
00000000  2
00000000  2  PROCEDURE GOVERNOR(INP : PUINT; VAR OUT : PUINT);
00000000  3
00000000  3     { Proportional controller with P = 30
00000000  3       causes rapid torque increase.        }
00000000  3
00000000  3  CONST    P = 30;           { SLOPE = 30 }
00000000  3           MAX = 16383;      { = 1.0 PU   }
00000000  3           MIN = 0;
00000000  3           MAX_30 = 546;   {  = 1/30 PU }
00000444  3
00000444  3  BEGIN    IF INP > MAX_30 THEN
00000450  3                OUT := MAX          { reached upper limit }
0000045C  3           ELSE
0000045C  3                IF INP > MIN THEN
00000464  3                     OUT := P * INP    { proportional range }
00000476  3                ELSE
00000476  3                     OUT := MIN;    { reached lower limit  }
0000047C  3  { OUT := MAX;      FOR TEST PURPOSES }
0000047C  3
0000047C  3  END; { GOVERNOR }
00000000  2
00000000  2
00000000  2
00000000  2  PROCEDURE TFIX(INP,TDSET,TURBSETD : PUINT; VAR OUT : PUINT);
00000000  3
00000000  3     { The torque is limited by the ceiling torque
00000000  3       value which is the lesser of either TDSET (which
00000000  3       equals (PSET + 400kW)/NSET  )  or TURBSETD which
00000000  3       is the available torque from the turbocharger.    }
00000000  3
00000000  3  CONST    MIN = -819;  { = - 0.05 PU     }
00000000  3
00000000  3  VAR      TORQUE : PUINT;
00000484  3
00000484  3  BEGIN
00000484  3      IF TURBSETD < TDSET THEN
00000490  3           TORQUE := TURBSETD
0000049B  3      ELSE
0000049B  3           TORQUE := TDSET;
0000049E  3
0000049E  3      IF INP > TORQUE THEN
```

```
402 000004AA  3           OUT := TORQUE
403 000004B6  3      ELSE
404 000004B6  3           IF INP > MIN THEN
405 000004C2  3                OUT := INP
406 000004CC  3           ELSE
407 000004CC  3                OUT := MIN;
408 000004D4  3
409 000004D4  3  END; { TFIX }
410 00000000  2
411 00000000  2
412 00000000  2
413 00000000  2  PROCEDURE SHUTDOWN(INP : PUINT; VAR OUT : PUINT);

414 00000000  3
415 00000000  3      { Max torque overloads the integrator to simulate
416 00000000  3        stalling during stall range.  Restart can occur
417 00000000  3        only if idling ( NOTCH 1) and "START" button is
418 00000000  3        held down.  A  0.1 PU torque is applied to hasten
419 00000000  3        the start-up process.                            }
420 00000000  3
421 00000000  3  CONST    MINOP = 4915;  { = 0.3 PU   }
422 00000000  3           MAX = 16383;   { = 1.0 PU     }
423 00000000  3           LKSTRT = 1638;  {= 0.1 PU     }
424 000004DC  3
425 000004DC  3  BEGIN
426 000004DC  3      IF (INP >= MINOP) AND (NOT STRTHYS) THEN
427 000004F8  3           OUT := 0         { no start-up effects  }
428 00000502  3      ELSE
429 00000502  3           IF NTCH1FLG AND STRTFLG THEN
430 0000050E  3                BEGIN
431 0000050E  3                     STRTHYS := TRUE;
432 00000514  3                     OUT := - LKSTRT;    { assist start-up }
433 0000051C  3                END
434 00000520  3           ELSE
435 00000520  3                OUT := MAX;      { apply overload torque }
436 00000528  3
437 00000528  3      IF INP > MINOP THEN
438 00000534  3           DLED := TRUE      { activate diesel monitor LED }
439 0000053E  3      ELSE
440 0000053E  3           DLED := FALSE;    { deactivate }
441 00000542  3
442 00000542  3  END; { SHUTDOWN }
443 00000000  2
444 00000000  2
445 00000000  2
446 00000000  2  PROCEDURE NODED1SUM(IN1,IN2,IN3: PUINT; VAR OUT: PUINT);
447 00000000  3
448 00000000  3      { NODED1 has 3 summing points and thus may exceed
449 00000000  3        the 16 bit range.  Thus this summing is done at
450 00000000  3        the 32 bit level, then converted back to 16 bit. }
451 00000000  3
```

```
452 00000000  3  CONST    MAXPOS = 16383;     { = 1.0 PU }
453 00000000  3           MAXNEG = -16383;
454 00000000  3
455 00000000  3  VAR      LOUTD1,LOUTD2 : LONGINT;
456 0000054A  3
457 0000054A  3  BEGIN
458 0000054A  3      LOUTD2 := IN1 - IN2;
459 0000055C  3      LOUTD1 := LOUTD2 - IN3;
460 0000056B  3
461 0000056B  3      IF ABS(LOUTD1) > MAXPOS THEN
462 0000057A  3           OUT := MAXNEG    { value out of 16 bit range }
463 00000586  3      ELSE
464 00000586  3           OUT := INTEGER(LOUTD1);  { value in 16 bit range }
465 00000590  3  END; { NODED1SUM }
466 00000000  2
467 00000000  2
468 00000000  2
469 00000000  2  PROCEDURE DGA;
470 00000598  3
471 00000598  3  BEGIN
472 00000598  3      DATASET [1] := ADJ(TDSET);

473 000005AA  3      DATASET [2] := ADJ(NSETLIM);
474 000005BC  3      DATASET [3] := ADJ(TSHAFT);
475 000005CE  3      DATASET [4] := ADJ(LDFCTR);
476 000005E2  3      DATASET [5] := ADJ(PLOAD);
477 000005F4  3      DATASET [6] := ADJ(TDLOAD);
478 00000606  3      DATASET [7] := ADJ(NDIESEL);
479 0000061A  3  END; { DGA }
480 00000000  2
481 00000000  2
482 00000000  2
483 00000000  2  PROCEDURE DGB;
484 00000622  3
485 00000622  3  BEGIN
486 00000622  3      DATASET [1] := ADJ(TDSET);
487 00000634  3      DATASET [2] := ADJ(TREF);
488 00000646  3      DATASET [3] := ADJ(TSHAFT);
489 0000065B  3      DATASET [4] := ADJ(TURBOTOR);
490 0000066A  3      DATASET [5] := ADJ(TURBSET);
491 0000067B  3      DATASET [6] := ADJ(TDELAY);
492 00000686  3      DATASET [7] := ADJ(NDIESEL);
493 0000069A  3  END; { DGB }
494 00000000  2
495 00000000  2
496 00000000  2
497 00000000  2  PROCEDURE DGC;
498 000006A2  3
499 000006A2  3  BEGIN
500 000006A2  3      DATASET [1] := ADJ(DLOUT);
501 000006B4  3      DATASET [2] := ADJ(SSOUT);
```

```
502 000006C6  3      DATASET [3] := ADJ(TSHAFT);
503 000006D8  3      DATASET [4] := ADJ(PLOAD);
504 000006EA  3      DATASET [5] := ADJ(TDLOAD);
505 000006FC  3      DATASET [6] := ADJ(LDFCTR);
506 00000710  3      DATASET [7] := ADJ(NDIESEL);
507 00000724  3  END; { DGC }
508 00000000  2
509 00000000  2
510 00000000  2
511 00000000  2  PROCEDURE DGIA;
512 0000072C  3
513 0000072C  3  BEGIN
514 0000072C  3      DATASET [1] := ADJ(LDFCTR);
515 00000740  3      DATASET [2] := ADJ(TDLOAD);
516 00000752  3
517 00000752  3
518 00000752  3      DATASET [5] := ADJ(PLOAD);
519 00000764  3
520 00000764  3      DATASET [7] := ADJ(NDIESEL);
521 00000778  3  END; { DGIA }
522 00000780  2
523 00000780  2
524 00000780  2
525 00000780  2
526 00000780  2  BEGIN { DIESEL }
527 00000780  2
528 00000780  2
529 00000780  2          PLANT(NODED1,NODED0);    { diesel plant = 1/S }
530 00000796  2
531 00000796  2          CHARAC(NODED0,NDIESEL);  { limit/shape output speed }

532 000007AE  2
533 000007AE  2          DLOSSES(NDIESEL,DLOUT);  { calculate losses  }
534 000007C6  2
535 000007C6  2
536 000007C6  2
537 000007C6  2     {    dN / dT limiter  prevents the speed setting -
538 000007C6  2          NSET from changing faster than .1 PU / sec.        }
539 000007C6  2
540 000007C6  2          DNDTINTGR(NODED2,NSETLIM);
541 000007DC  2
542 000007DC  2          NODED3 := NSET - NSETLIM;
543 000007E8  2
544 000007E8  2          DNDTCHAR(NODED3,NODED2);
545 000007FC  2
546 000007FC  2
547 000007FC  2
548 000007FC  2          NERR := NSETLIM - NDIESEL;    { delta speed signal }
549 0000080A  2
550 0000080A  2          GOVERNOR(NERR,TABS);     { prop. controller P = 50 }
551 0000081E  2
```

```
0000081E  2          TREF := TABS - DLOUT;    { subtract diesel losses }
0000082A  2
0000082A  2          TFIX(TREF,TDSET,TDELAY,TSHAFT);   { torque limiter }
00000846  2                  {    ^^^^^   }
00000846  2                  {    !!!!!   }
00000846  2                { TDELAY <=> 16383  }
00000846  2     {    Turbocharger produces a gradual build up of
00000846  2          requested torque as a function of present torque
00000846  2          and present diesel speed.                         }
00000846  2
00000846  2          TGEN(TSHAFT,NSET,TURBOTOR);
00000860  2
00000860  2          TURBOFIX(TURBOTOR,TDSET,TURBSET);
0000087A  2
0000087A  2          TSD(TURBSET,TDELAY);
00000890  2
00000890  2
00000890  2     {    A load factor between 0 and 1 is determined
00000890  2          from TREF and TSHAFT.  When TSHAFT exceeds
00000890  2          TREF by more than 3 % the load factor begins
00000890  2          to decrease from 1.                               }
00000890  2
00000890  2          TSWITCH(NERR,TREF,TSHAFT,LFIN);
000008AE  2
000008AE  2          LFCHAR(LFIN,LF1);
000008C4  2
000008C4  2          LFINTGR(LF1,LDFCTR);
000008DC  2
000008DC  2
000008DC  2
000008DC  2
000008DC  2
000008DC  2          SHUTDOWN(NDIESEL,SSOUT);         { see if in STALL mode }
000008F4  2
000008F4  2
000008F4  2          IF JOIN THEN   { train load is connected }
000008FC  2               IF STRTFLG AND NTCH1FLG THEN
00000908  2                    TDLOAD := 0   { so that the diesel may start up }
00000910  2               ELSE

00000910  2                   TDLOAD := PUDIV(PLOAD,NDIESEL)
00000928  2          ELSE  { pot load connected }
00000928  2               TDLOAD := TEST;  { TEST <=> 0 }
0000092E  2
0000092E  2               { NODED1 requires special consideration due
0000092E  2                 to the fact that the resulting sum may
0000092E  2                 overflow the 16 bit range.              }
0000092E  2
0000092E  2          NODED1SUM(TSHAFT,TDLOAD,SSOUT,NODED1);
0000094C  2
0000094C  2          IF SELECT = 'A' THEN DGA;
```

```
 602 00000962  2           IF SELECT = 'B' THEN DGB;
 603 0000097B  2           IF SELECT = 'C' THEN DGC;
 604 0000098E  2           IF SELECT = 'I' THEN DGIA;
 605 000009A4  2
 606 000009A4  2           IF SELECT = 'T' THEN
 607 000009B0  2               DATASET [7] := ADJ(NDIESEL);
 608 000009C0  2
 609 000009C0  2  END; { DIESEL }
 610 00000000  1
 611 00000000  1
 612 00000000  1  $GLOBPROC OFF$
 613 00000000  1
 614 00000000  1  .   { main code in main module LOCO }

End of compilation, number of errors=    0
```

Program ELOCO

Procedure LSCINTGR (INPSC φ, WREQ : PUINT ; OUTSC φ : PUINT)
Locomotive speed control integrator block performs the function $H(s) = 1/15s$.

Procedure LSCLIM (INP, LIM : PUINT ; VAR OUT : PUINT)
The minimum power request is kept above .001 PU to avoid negative values.

Procedure PECHAR (INP : PUINT ; VAR OUT : PUINT)
Proportional controller with P = 50 used in conjunction with PE INTGR to provide a d/dt limiter that limits the rate of change of the W power request.

Procedure PE INTGR (INPPE φ : PUINT ; VAR OUTPE φ : PUINT)
$H(s) = 1/1+0.25$ used to limit d/dt of W (see PECHAR).

Procedure VCHAR (INP, DSPEED : PUINT ; VAR OUT : PUINT)
Torque conditioning characteristic limits torque value as a function of DC voltage produced by the diesel (which is in turn a function of the diesel speed).

Procedure TGEN (P : PUINT ; VAR T : PUINT)
The power request W is converted to a torque request according to the defined MICAS table of request values. Intermediate values are interpolated.

Procedure PNTCHAR (P, N, T : PUINT ; VAR TOUT : PUINT)
A set of power curves are created as a function of torque and speed. The relationship is linear, i.e. $T = P/n$, for $T < Trequest$ (generated by TGEN). For $T \geq Trequest$, $T = Trequest$.

Procedure TORQINN (INP : PUINT ; VAR OUT : PUINT) ;
If the locomotive switches into "braking", the torque is inverted (i.e. negative).

Procedure DGD data set procedures
DGE assign various internal
DGI values to real time
DGT display system.

```
 1 00000000  1  "68000"
 2 00000000  1  $EXTENSIONS$
 3 00000000  1
 4 00000000  1  PROGRAM ELOCO;
 5 00000000  1
 6 00000000  1
 7 00000000  1     {     ELOCO is an external module containing the procedures
 8 00000000  1           associated with the torque command control section of
 9 00000000  1           the locomotive simulation.
10 00000000  1                The EXTERNAL variables listed below represent
11 00000000  1           globals from the main program module which are used
12 00000000  1           within this module.
13 00000000  1                                                                 }
14 00000000  1
15 00000000  1  CONST     N = 8;
16 00000000  1            ScaleP = 13334;   { 0.8138 PU }
17 00000000  1
18 00000000  1  TYPE      PUINT = SIGNED_16;
19 00000000  1            INTEGER = SIGNED_16;
20 00000000  1            BYTE = SIGNED_8;
21 00000000  1            EIGHT = UNSIGNED_8;
22 00000000  1            LPUINT = SIGNED_32;
23 00000000  1            LONGINT = SIGNED_32;
24 00000000  1            STATUSBITS = SET OF 0..7;
25 00000000  1            PROFILE = ARRAY [0..255] OF INTEGER;
26 00000000  1            TABLE = ARRAY [1..5] OF PROFILE;
27 00000000  1
28 00000000  1
29 00000000  1  $EXTVAR ON$
30 00000000  1
31 00000000  1  VAR       OUTPE1,INPPE1 : PUINT;
32 00000000  1            PEINIT : PUINT;
33 00000000  1            INPW1,OUTW1 : PUINT;
34 00000000  1            WINIT : PUINT;
35 00000000  1            OUTSC1,INPSC1 : PUINT;
36 00000000  1            LSCTIME : INTEGER;
37 00000000  1            SELECT : CHAR ;
38 00000000  1            MOTORING,BRAKING : BOOLEAN;
39 00000000  1            INTSLT,EXTSLT : BOOLEAN;
40 00000000  1            LFIN : BOOLEAN;
41 00000000  1            TROOP : BOOLEAN;
42 00000000  1            DATASET : ARRAY [1..N] OF EIGHT;
43 00000000  1
44 00000000  1  $EXTVAR OFF$
45 00000000  1
46 00000000  1            NDIFF : PUINT;   { global }
47 00000000  1
48 00000000  1
49 00000000  1
50 00000000  1  FUNCTION PUMULT(XER,XAND : PUINT): PUINT; EXTERNAL;
```

```
 51 00000000  1
 52 00000000  1  FUNCTION PUDIV(DVND,DVSR : PUINT): PUINT; EXTERNAL;
 53 00000000  1
 54 00000000  1  FUNCTION ADJ(SIXTN : PUINT): EIGHT; EXTERNAL;
 55 00000000  1
 56 00000000  1
 57 00000000  1
 58 00000000  1
 59 00000000  1  $GLOBPROC ON$

 60 00000000  1
 61 00000000  1
 62 00000000  1  PROCEDURE ELECDRIVE(    TEST,PESET,NDIESEL,PSET,NMOT,
 63 00000000  2                          TMOT,LDFCTR,NLOCO,NLOCOSET : PUINT;
 64 00000000  2                      VAR TEREF,DLOAD : PUINT);
 65 00000000  2
 66 00000000  2
 67 00000000  2  VAR      PESETD,PERR : PUINT;
 68 00000004  2           LFPESET : PUINT;
 69 00000006  2           TLIM : PUINT;
 70 00000008  2           PEV,TAV : PUINT;
 71 0000000C  2           PMOT : PUINT;
 72 0000000E  2           NBREF : PUINT;
 73 00000010  2           SCPESET,NLOCOERR : PUINT;
 74 00000000  2
 75 00000000  2
 76 00000000  2
 77 00000000  2
 78 00000000  2
 79 00000000  2     PROCEDURE LSCINTGR(INPSCO,WREQ : PUINT; VAR OUTSCO : PUINT);
 80 00000000  3
 81 00000000  3     {    60 s integrator used in locomotive speed control
 82 00000000  3          loop.
 83 00000000  3
 84 00000000  3                    OUTSCO  =    1
 85 00000000  3                    ------    --------
 86 00000000  3                    INPSCO      (60)S                      }
 87 00000000  3
 88 00000000  3     CONST      A = 205;   { 0.0125 PU }
 89 00000000  3                K = 8;     { prop. constant }
 90 00000000  3
 91 00000000  3                AbsMax = 15973;  { 2**14 - 1 - 2A  }
 92 00000000  3                AbsMin = 0;      { 0.0 PU }
 93 00000000  3                IntFctr = 15;  { sample time scaling }
 94 00000000  3
 95 00000000  3     VAR        ThOp,Limit : PUINT;
 96 00000004  3
 97 00000004  3     BEGIN
 98 00000004  3          LSCTIME := LSCTIME + 1;
 99 00000008  3          IF LSCTIME >= IntFctr THEN
100 00000014  3          BEGIN
```

```
101 00000014  3              LSCTIME := 0;
102 00000018  3
103 00000018  3
104 00000018  3              ThOp := OUTSC1 + K * PUMULT(A,INPSC0)
105 00000018  3                      + K * PUMULT(A,INPSC1)
106 00000018  3                      - K * INPSC1 + K * INPSC0;
107 00000064  3
108 00000064  3              IF WREQ > AbsMax THEN
109 00000070  3                  Limit := AbsMax
110 0000007A  3              ELSE
111 0000007A  3                  Limit := WREQ;
112 00000080  3
113 00000080  3              IF ThOp > Limit THEN
114 0000008C  3                  OUTSC0 := Limit
115 0000009B  3              ELSE
116 0000009B  3                  IF ThOp < AbsMin THEN
117 000000A0  3                      OUTSC0 := AbsMin
118 000000AA  3                  ELSE

119 000000AA  3                      OUTSC0 := ThOp;
120 000000B2  3
121 000000B2  3              OUTSC1 := OUTSC0;
122 000000BA  3              INPSC1 := INPSC0;
123 000000C0  3          END;
124 000000C0  3
125 000000C0  3      END; { LSCINTR }
126 00000000  2
127 00000000  2
128 00000000  2
129 00000000  2      PROCEDURE LSCLIM(INP,LIM : PUINT; VAR OUT : PUINT);
130 00000000  3
131 00000000  3      {   The incoming power request is limited by the
132 00000000  3          output of LSCINTGR.                               }
133 00000000  3
134 00000000  3      CONST    MIN = 16;       { .001 PU }
135 000000C8  3
136 000000C8  3      BEGIN
137 000000C8  3
138 000000C8  3          IF INP > LIM THEN
139 000000D4  3              OUT := LIM
140 000000E0  3          ELSE
141 000000E0  3              IF INP > MIN THEN
142 000000EC  3                  OUT := INP
143 000000F6  3              ELSE
144 000000F6  3                  OUT := MIN;
145 000000FE  3
146 000000FE  3      END; { LSCLIM }
147 00000000  2
148 00000000  2
149 00000000  2
150 00000000  2      PROCEDURE PECHAR(INP : PUINT; VAR OUT : PUINT);
```

```
151 00000000  3
152 00000000  3      {    Proportional controller with P = 50 provides
153 00000000  3           an upper and lower integrating input value.   }
154 00000000  3
155 00000000  3      CONST     P = 30;
156 00000000  3                MAX = 16383;
157 00000000  3                MAX_50 = 546;      { MAX/30 }
158 00000106  3
159 00000106  3      BEGIN
160 00000106  3           IF INP > MAX_50 THEN
161 00000112  3                OUT := MAX     { go to upper limit }
162 0000011E  3           ELSE
163 0000011E  3                IF INP >= (- MAX_50) THEN
164 0000012A  3                     OUT := P * INP      { proportional region }
165 00000138  3                ELSE
166 00000138  3                     OUT := - MAX;       { go to lower limit }
167 00000140  3      END; { PECHAR }
168 00000000  2
169 00000000  2
170 00000000  2
171 00000000  2      PROCEDURE PEINTGR(INPPEO : PUINT; VAR OUTPEO : PUINT);
172 00000000  3
173 00000000  3      {    Performs the first order delay function:
174 00000000  3
175 00000000  3                          OUTPEO   =      1
176 00000000  3                          -------     -------------
177 00000000  3                          INPPEO      1 + (200ms)S }

178 00000000  3
179 00000000  3      CONST     A = 14454;
180 00000000  3                B = 965;
181 00000000  3                AbsMax = 16383;
182 00000000  3                AbsMin = -16383;    { - 1.0 PU }
183 00000148  3
184 00000148  3
185 00000148  3      BEGIN
186 00000148  3           OUTPEO := PUMULT(A,OUTPE1) + PUMULT(B,INPPEO)
187 00000148  3                + PUMULT(B,INPPE1);
188 0000018C  3
189 0000018C  3           IF OUTPEO > AbsMax THEN
190 0000019A  3                OUTPEO := AbsMax;
191 000001A2  3           IF OUTPEO < AbsMin THEN
192 000001B0  3                OUTPEO := AbsMin;
193 000001B8  3
194 000001B8  3           OUTPE1 := OUTPEO;
195 000001C0  3           INPPE1 := INPPEO;
196 000001C6  3
197 000001C6  3      END; { PEINTGR }
198 00000000  2
199 00000000  2
200 00000000  2
```

```
00000000  2      PROCEDURE VCHAR(INP,DSPEED : PUINT; VAR OUT : PUINT);
00000000  3
00000000  3      (    The available drive torque is limited by the DC
00000000  3           voltage produced by the diesel.                }
00000000  3
00000000  3      CONST     M = 6144;       ( 0.3750 PU )
00000000  3                B = 0;          ( 0.0000 PU )
00000000  3
00000000  3                NBRKH = 11916; ( 0.73 PU, N5 < 800 RPM < N6 )
00000000  3                NBRKL = 5958;  ( 0.36 PU = 400 RPM = N1 )
00000000  3                NMIN = 1638;   ( 0.1 PU )
00000000  3
00000000  3                MAXV = 16383;    ( 1000 V )
00000000  3                MINV = 8192;     ( 500 V )
00000000  3                ZERV = 0;        ( 0 V )
00000000  3
00000000  3                MIN = 0;
00000000  3
00000000  3      VAR       V : PUINT;
00000002  3                TL : PUINT;
000001CE  3
000001CE  3      BEGIN
000001CE  3           IF DSPEED < NMIN THEN
000001DA  3                V := ZERV
000001E2  3           ELSE
000001E2  3                IF DSPEED < NBRKL THEN
000001EE  3                     V := MINV
000001F8  3                ELSE
000001F8  3                     IF DSPEED < NBRKH THEN
00000204  3                          V := (PUMULT(M,DSPEED) - B) + DSPEED
00000220  3                     ELSE
00000220  3                          V := MAXV;
00000226  3           TL := V;
0000022C  3           IF INP > TL THEN
00000238  3                OUT := TL
00000244  3           ELSE

00000244  3                IF INP > MIN THEN
0000024C  3                     OUT := INP
00000258  3                ELSE
00000258  3                     OUT := MIN;
0000025E  3
0000025E  3
0000025E  3      END; ( VCHAR )
00000000  2
00000000  2
00000000  2
00000000  2      PROCEDURE TGEN(P : PUINT; VAR T : PUINT);
00000000  3
00000000  3      (    This procedure outputs a torque value
00000000  3           as a function of the input power value
```

```
 251 00000000  3          according to the predefined discrete
 252 00000000  3          relationship.  Values in between the
 253 00000000  3          defined points are interpolated.      }
 254 00000000  3
 255 00000000  3     TYPE      VALUES = ARRAY [0..9] OF PUINT;
 256 00000000  3
 257 00000000  3     VAR       PBASE : VALUES;
 258 00000014  3               TBASE : VALUES;
 259 00000028  3               RMDR : VALUES;
 260 0000003C  3               I,TMULT : INTEGER;
 261 00000040  3               PINC : PUINT;
 262 00000266  3
 263 00000266  3     BEGIN
 264 00000266  3          PBASE [0] :=     0;
 265 0000026A  3          PBASE [1] :=   164;
 266 00000270  3          PBASE [2] :=   460;
 267 00000276  3          PBASE [3] :=  1055;
 268 0000027C  3          PBASE [4] :=  1845;
 269 00000282  3          PBASE [5] :=  3418;
 270 00000288  3          PBASE [6] :=  5912;
 271 0000028E  3          PBASE [7] :=  8865;
 272 00000294  3          PBASE [8] := 13334;
 273 0000029A  3          PBASE [9] := 16383;
 274 000002A0  3
 275 000002A0  3          I := 0;
 276 000002A4  3
 277 000002A4  3          WHILE P >= PBASE [I] DO
 278 000002BA  3               I := I + 1;
 279 000002C0  3
 280 000002C0  3          I := I - 1;
 281 000002C4  3          PINC := P - PBASE [I];
 282 000002DA  3
 283 000002DA  3               CASE I OF
 284 000002E4  3
 285 000002E4  3          0 :  T := 0 + 11 * PINC + PUMULT(8894,PINC);
 286 00000312  3          1 :  T := 1888 + 6 * PINC + PUMULT(5939,PINC);
 287 00000340  3          2 :  T := 3775 + 3 * PINC + PUMULT(2891,PINC);
 288 0000036E  3          3 :  T := 5663 + 2 * PINC + PUMULT(6338,PINC);
 289 0000039C  3          4 :  T := 7551 + PINC + PUMULT(3280,PINC);
 290 000003BE  3          5 :  T := 9439 + PUMULT(12405,PINC);
 291 000003DA  3          6 :  T := 11326 + PUMULT(10472,PINC);
 292 000003F6  3          7 :  T := 13214 + PUMULT(6922,PINC);
 293 00000412  3
 294 00000412  3          OTHERWISE T := 15102;
 295 0000041A  3

 296 0000041A  3               END { Case }
 297 00000448  3
 298 00000448  3     END; { TGEN }
 299 00000000  2
 300 00000000  2
```

```
00000000  2
00000000  2     PROCEDURE PNTCHAR(P,T,N : PUINT; VAR TOUT : PUINT);
00000000  3
00000000  3     CONST     M = 6447;    { 0.3935 PU }
00000000  3               PdToPm = 4;
00000000  3               B = 15102;       { 0.9218 PU = 100% TE }
00000000  3               DeltaZero = 164;  { 0.01 PU }
00000000  3               AbsMax = 16383;   { 1.0 PU }
00000000  3               Hyst = 328;    { 0.02 PU }
00000000  3
00000000  3     VAR       TOUT1,TOUT2 : PUINT;
00000004  3               TMAX,TSCALE : PUINT;
00000008  3               PMS : PUINT;
00000450  3
00000450  3
00000450  3     BEGIN
00000450  3          IF N < DeltaZero THEN       { let limits rule for }
0000045C  3               BEGIN
0000045C  3                 TOUT1 := AbsMax;       { N (denom.) close to 0 }
00000462  3                 N := DeltaZero;
00000468  3               END
0000046C  3          ELSE
0000046C  3               BEGIN
0000046C  3                    PMS := P / PdToPm;
0000047A  3                    TOUT1 := PUDIV(PMS,N);
0000048A  3               END;
0000048A  3
0000048A  3          TMAX := B - PUMULT(M,N);
000004A2  3          IF T > TMAX THEN
000004AC  3               TOUT2 := TMAX
000004B4  3          ELSE
000004B4  3               TOUT2 := T;
000004BA  3
000004BA  3          IF TOUT1 < TOUT2 THEN
000004C6  3               BEGIN     { power operation }
000004C6  3                    TOUT := TOUT1;
000004CC  3                    IF (TOUT2 - TOUT1) > Hyst THEN
000004DA  3                         TRQOP := FALSE;
000004DE  3               END
000004E2  3          ELSE
000004E2  3               BEGIN     { torque operation }
000004E2  3                    TOUT := TOUT2;
000004EA  3                    TRQOP := TRUE;
000004F0  3               END;
000004F0  3
000004F0  3     END; { PNTCHAR }
00000000  2
00000000  2
00000000  2
00000000  2     PROCEDURE TORQINV(INP : PUINT; VAR OUT : PUINT);
000004F8  3
```

```
352 000004F8  3      BEGIN
353 000004F8  3          IF BRAKING AND (NOT MOTORING) THEN
354 0000050A  3              OUT := - INP

355 0000051A  3          ELSE
356 0000051A  3              OUT := INP;
357 00000522  3      END; ( TORQINV )
358 00000000  2
359 00000000  2
360 00000000  2
361 00000000  2      PROCEDURE DGD;
362 0000052A  3
363 0000052A  3      BEGIN
364 0000052A  3          DATASET [1] := ADJ(PESETD);
365 0000053C  3          DATASET [2] := ADJ(PEV);
366 0000054E  3          DATASET [3] := ADJ(TLIM);
367 00000560  3          DATASET [4] := ADJ(TEREF);
368 00000574  3          DATASET [5] := ADJ(DLOAD);
369 00000588  3          DATASET [6] := ADJ(NMOT);
370 0000059A  3          DATASET [7] := ADJ(NDIESEL);
371 000005AC  3      END; ( DGD )
372 00000000  2
373 00000000  2
374 00000000  2
375 00000000  2      PROCEDURE DGE;
376 000005B4  3
377 000005B4  3      BEGIN
378 000005B4  3          DATASET [1] := ADJ(PESET);
379 000005C6  3          DATASET [2] := ADJ(PESETD);
380 000005D8  3          DATASET [3] := ADJ(PEV);
381 000005EA  3          DATASET [4] := ADJ(TAV);
382 000005FC  3          DATASET [5] := ADJ(NLOCOSET);    ( NLOCOSET (=) TLIM )
383 0000060E  3          DATASET [6] := ADJ(NDIFF);   ( NDIFF (=) TEREF )
384 0000061C  3          DATASET [7] := ADJ(SCPESET); ( SCPESET (=) NDIESEL )
385 0000062E  3
386 0000062E  3      END; ( DGE )
387 00000000  2
388 00000000  2
389 00000000  2
390 00000000  2  PROCEDURE DGI;
391 00000636  3
392 00000636  3      BEGIN
393 00000636  3
394 00000636  3
395 00000636  3
396 00000636  3
397 00000636  3
398 00000636  3          DATASET [6] := ADJ(NMOT);
399 0000064B  3
400 0000064B  3      END; ( DGI )
401 00000000  2
```

```
00000000  2
00000000  2  PROCEDURE DGT;
00000650  3
00000650  3      BEGIN
00000650  3          DATASET [1] := ADJ(TMOT);
00000662  3
00000662  3
00000662  3
00000662  3
00000662  3          DATASET [6] := ADJ(NMOT);
00000674  3          DATASET [7] := ADJ(NDIESEL);
00000686  3      END; { DGTB }

0000068E  2
0000068E  2
0000068E  2
0000068E  2  BEGIN { ELECDRIVE }
0000068E  2
0000068E  2      {   A locomotive speed control loop controls the
0000068E  2          locomotive speed, NLOCO.                         }
0000068E  2
0000068E  2      NLOCOERR := NLOCOSET - NLOCO;
0000069A  2
0000069A  2      LSCINTGR(NLOCOERR,PESET,NDIFF);     { NLOCOSET <=> NLOCOERR }
000006B2  2
000006B2  2      LSCLIM(PESET,NDIFF,SCPESET);    { 16383 <=> NDIFF }
000006CC  2
000006CC  2
000006CC  2      {   The load factor signal from the diesel (DLOCO)
000006CC  2          can be switched in to limit the power request.   }
000006CC  2
000006CC  2      IF LFIN THEN
000006D4  2          LFPESET := PUMULT(LDFCTR,SCPESET)
000006EA  2      ELSE
000006EA  2          LFPESET := SCPESET;
000006F0  2
000006F0  2
000006F0  2
000006F0  2      {   dPESET / dt limiter limits change in
000006F0  2          PESET to 100 % / 0.3 sec.              }
000006F0  2
000006F0  2      PEINTGR(LFPESET,PESETD);
00000706  2
00000706  2
00000706  2          {   The power request is scaled according to
00000706  2              the PU base value.                 }
00000706  2
00000706  2      PEV := PUMULT(ScaleP,PESETD);
00000718  2
00000718  2
00000718  2          {   Convert the power request to a torque
00000718  2               value so that torque limiting may be
00000718  2               implemented.                    }
00000718  2
00000718  2      TGEN(PEV,TAV);
0000072E  2
0000072E  2
0000072E  2      { The torque value TAV is limited by  the DC
0000072E  2        voltage at the diesel output.              }
0000072E  2
0000072E  2      VCHAR(TAV,NDIESEL,TLIM);
00000748  2
00000748  2
00000748  2
00000748  2      IF INTSLT OR EXTSLT THEN       { use TE request to load }
00000754  2                                     { diesel during SLT      }
00000754  2          BEGIN               { and ground out motor drive. }
00000754  2              DLOAD := PEV;
0000075C  2              TLIM := 0;
00000760  2          END
00000764  2      ELSE          { connect TE to motor drive, calculate     }
00000764  2                    { motor power, scale to diesel reference,  }
00000764  2          BEGIN    { and output to diesel load.               }
00000764  2              PMOT := PUMULT(NMOT,TMOT);
00000776  2              DLOAD := 4 * PMOT;
00000784  2              IF DLOAD < 0 THEN
0000078E  2                  DLOAD := 0;
00000794  2          END;
00000794  2
00000794  2
00000794  2      PNTCHAR(PEV,TLIM,NMOT,MBREF);
000007B2  2
000007B2  2      TORQINV(MBREF,TEREF);     { MBREF <=> TEST }
000007CA  2
000007CA  2      IF SELECT = 'D' THEN DGD;
000007E0  2      IF SELECT = 'E' THEN DGE;
000007F6  2      IF SELECT = 'I' THEN DGI;
0000080C  2      IF SELECT = 'T' THEN DGT;
00000822  2
00000822  2
00000822  2  END; { ELECDRIVE }
00000000  1
00000000  1
00000000  1  $GLOBPROC OFF$
00000000  1
00000000  1  .   { main code in main program module }
00000000  1

End of compilation, number of errors=    0
```

Program MLOCO

Procedure MAXSEARCH (NMOT, NLOCO, TMOT, FSPOT : PUINT ; VAR SLOP : PUINT)

A successive comparison of torque algorithm that finds and tracks the maximum torque point in the slip-torque traction relationship.

Procedure ELDRDELAY (INPE φ : PUINT ; VAR OUTEO : PUINT)

1st order delay $H(s) = 1/ 1+(50ms)s$ provides a 50ms delay that simulates the electrical transient of the drive section.

Procedure ACDRIVE (INP, NM, NL, SLOP : PUINT ; VAR OUT : PUINT)

The generation of torque in an AC drive is simulated.

Procedure MTRPLANT (INPM φ : PUINT ; VAR OUTM φ : PUINT)

The motors are modelled using a 1s integrator ($H(s) = 1/ 1s$).

Procedure MLOSSES (INP : PUINT ; VAR OUT : PUINT)

Losses in the motor drive section are calculated as a function of motor speed (NMOT).

Procedure RAMPCHAR (QI1, QO2 : PUINT ; VAR QI, QO : PUINT)

This procedure changes the two multiplier values used to modify the traction torque/slip characteristic. When 's' is typed on the keyboard, the multiplier values "ramp" with a fixed time from their present value to the new setting on the external potentiometer.

Procedure FCC (GSlo : INTEGER ; GLin, GnLin : PUINT ; VAR FCProf : LIST ; Inp : PUINT ; VAR OUT : POINT)

The input slip value is conditioned as a pointer to the traction torque/slip table and a torque value is obtained based on the programmed characteristic.

~~Procedure SHUTDOWN (INP : PUINT ; VAR OUT : PUINT)~~
~~Enforces that diesel can only be started in notch 1. Torque is added to speed start up and a negative torque is added to stall the diesel when speed drops below 0.3 PU.~~

~~Procedure NODED1SUM (IN1, IN2, IN3 : PUNIT ; VAR OUT : PUINT)~~
~~The particular node has three signals being summed. To avoid overflow, summing is done double precision and then truncated.~~

~~Procedure DGA; data set procedures~~
~~DGB assign various internal~~
~~DGC values to real time~~
~~DGIA display system.~~

```
00000000  1  "68000"
00000000  1  $EXTENSIONS$
00000000  1
00000000  1  PROGRAM MLOCO;
00000000  1
00000000  1
00000000  1      {    MLOCO is an external module containing the procedure
00000000  1           MOTORS which in turn contains all the procedures
00000000  1           for the inverter, motor and traction force sections
00000000  1           of the simulation.
00000000  1                The EXTERNAL variables listed below represent
00000000  1           globals from the main program module which are used
00000000  1           within this module.
00000000  1                                                                }
00000000  1
00000000  1  CONST    N = 8;
00000000  1           LENGTH = 255;
00000000  1
00000000  1  TYPE     PUINT = SIGNED_16;
00000000  1           INTEGER = SIGNED_16;
00000000  1           BYTE = SIGNED_8;
00000000  1           EIGHT = UNSIGNED_8;
00000000  1           LPUINT = SIGNED_32;
00000000  1           LONGINT = SIGNED_32;
00000000  1           STATUSBITS = SET OF 0..7;
00000000  1           LIST = ARRAY [- LENGTH..(LENGTH - 1)] OF PUINT;
00000000  1
00000000  1           PROFILE = RECORD
00000000  2                           SLOPE : INTEGER;
00000000  2                           LIN : PUINT;
00000000  2                           NLIN : PUINT;
00000000  2                           TABLE : LIST;
00000000  1                      END;
00000000  1
00000000  1  VAR      STEPCOUNT : INTEGER;
00000002  1           SMPLCNT : INTEGER;
00000004  1           INCI,INCO : PUINT;
00000008  1           FINT : PUINT;
0000000A  1
0000000A  1
0000000A  1  $EXTVAR ON$
0000000A  1           LOCALTIME : INTEGER;
0000000A  1           NSEA : BOOLEAN;
0000000A  1           TMOTOLD : PUINT;
0000000A  1           TROOP : BOOLEAN;
0000000A  1           MSCIN : BOOLEAN;
0000000A  1
0000000A  1           OUTE1,INPE1 : PUINT;
0000000A  1           OUTLD1,INPLD1 : PUINT;
0000000A  1           INPM1 : PUINT;
```

```
0000000A  1           OUTM1 : LPUINT;
0000000A  1           NMREF,F1 : PUINT;
0000000A  1           DATASET : ARRAY [1..N] OF EIGHT;
0000000A  1           LMOT,SLIP : PUINT;
0000000A  1           TEREFD : PUINT;
0000000A  1  NMLIM,F2 : PUINT;
0000000A  1
0000000A  1           RAMPSTART,RAMPRUN : BOOLEAN;
0000000A  1           MPRI,MPRO : PUINT;

0000000A  1           KFCTR : INTEGER;
0000000A  1
0000000A  1           CURVESET : ARRAY [1..3] OF PROFILE;
0000000A  1
0000000A  1  $EXTVAR OFF$
00000000  1
00000000  1
00000000  1  FUNCTION PUMULT(XER,XAND : PUINT): PUINT; EXTERNAL;
00000000  1
00000000  1  FUNCTION PUDIV(DVND,DVSR : PUINT): PUINT; EXTERNAL;
00000000  1
00000000  1  FUNCTION ADJ(SXTN : PUINT): EIGHT; EXTERNAL;
00000000  1
00000000  1  FUNCTION LPUMULT(XER,XAND : PUINT): LPUINT; EXTERNAL;
00000000  1
00000000  1
00000000  1
00000000  1  $GLOBPROC ON$
00000000  1
00000000  1  PROCEDURE MAXSEARCH(    NMOT,NLOCO,TMOT,FSPOT : PUINT;
00000000  2                      VAR SLOP : PUINT);
00000000  2
00000000  2  CONST    SMin = 410;    { .025 PU }
00000000  2           SlopMin = 458; { .028 PU }
00000000  2           TDMin = 10;   { .0015 PU }
00000000  2           SlStep = 66;  { .0040 PU }
00000000  2
00000000  2           TimeFctr = 10;       { Cycle time = 10 * 25ms }
00000000  2           KMax = 4905;         { 0.3 PU }
00000000  2
00000000  2  VAR      SLPOT : PUINT;
00000004  2
00000004  2  BEGIN
00000004  2
00000004  2     LOCALTIME := LOCALTIME + 1;
00000008  2
00000008  2     IF LOCALTIME >= TimeFctr THEN
00000014  2      BEGIN
00000014  2        SLPOT := PUMULT(KMax,FSPOT);
00000026  2        LOCALTIME := 0;
0000002A  2          IF MSCIN THEN
```

```
101 00000032  2            IF TROOP THEN
102 0000003A  2
103 0000003A  2               IF (NMOT - MLOCO) >= SMin THEN
104 0000004A  2
105 0000004A  2                    IF NSEA THEN
106 00000052  2                         BEGIN
107 00000052  2                              TMOTOLD := TMOT;
108 00000058  2                              SLOP := SLOP + SlStep;
109 00000064  2
110 00000064  2                              IF SLOP > SLPOT THEN
111 00000072  2                                   BEGIN
112 00000072  2                                        SLOP := SLOP - SlStep * 2;
113 0000007E  2
114 0000007E  2                                        IF SLOP <= SlopMin THEN
115 0000008C  2                                             SLOP := SlopMin;
116 00000094  2
117 00000094  2                                        NSEA := TRUE;
118 0000009A  2                                   END

119 0000009E  2                              ELSE
120 0000009E  2                                   NSEA := FALSE;
121 000000A2  2                         END
122 000000A6  2                    ELSE
123 000000A6  2                         BEGIN
124 000000A6  2                              IF (TMOT - TMOTOLD) < TDMin THEN
125 000000B6  2                                   BEGIN
126 000000B6  2                                        SLOP := SLOP - SlStep * 2;
127 000000C2  2
128 000000C2  2                                        IF SLOP <= SlopMin THEN
129 000000D0  2                                             SLOP := SlopMin;
130 000000D8  2                                   END;
131 000000D8  2
132 000000D8  2                              NSEA := TRUE;
133 000000DE  2                         END
134 000000E2  2               ELSE
135 000000E2  2                    BEGIN
136 000000E2  2                         SLOP := SlopMin;
137 000000EA  2                         NSEA := TRUE;
138 000000F0  2                    END
139 000000F4  2            ELSE
140 000000F4  2               BEGIN
141 000000F4  2                    SLOP := SLOP - 4 * SlStep;
142 00000100  2                    IF SLOP < SlopMin THEN
143 0000010E  2                         SLOP := SlopMin;
144 00000116  2                    NSEA := TRUE;
145 0000011C  2               END
146 00000120  2          ELSE
147 00000120  2               SLOP := SLPOT;
148 00000128  2      END;
149 00000128  2
150 00000128  2  END; { MAXSEARCH }
```

```
00000000  1
00000000  1
00000000  1
00000000  1
00000000  1  PROCEDURE MOTORS(     TEST,TEREF,MLOCO,TRGTI,TRGTO,SLOP : PUINT;
00000000  2                        PROFILENUM : INTEGER;
00000000  2                    VAR TMOT,NMOT,FLOCO : PUINT);
00000000  2
00000000  2  (   Simulates the traction induction motor drives     )
00000000  2
00000000  2  VAR      XSLIP,XFLOCO : PUINT;
00000004  2           TMOT1,MLOCOD : PUINT;
00000000  2
00000000  2
00000000  2
00000000  2      PROCEDURE ELDRDELAY(INPE0 : PUINT; VAR OUTE0: PUINT);
00000000  3
00000000  3      ( Time lag due to electric drive stage is simulated
00000000  3        using a 50 ms first order delay of the form:
00000000  3
00000000  3                    OUTE0    =      1
00000000  3                    ------      -------------
00000000  3                    INPE0       1 + (50ms)S              )
00000000  3
00000000  3      CONST     A = 16059;     ( 0.9802 PU )
00000000  3                B = 162;       ( 0.0099 PU )
00000000  3

00000000  3                ABSMAX = 16383;     ( 1.0 PU )
00000000  3                ABSMIN = -16383;    ( -1.0 PU )
00000130  3
00000130  3      BEGIN
00000130  3           OUTE0 := PUMULT(A,OUTE1) + PUMULT(B,INPE0)
00000130  3                   + PUMULT(B,INPE1);
00000174  3
00000174  3           IF OUTE0 > ABSMAX THEN
00000182  3                OUTE0 := ABSMAX;
0000018A  3           IF OUTE0 < ABSMIN THEN
00000198  3                OUTE0 := ABSMIN;
000001A0  3
000001A0  3           OUTE1 := OUTE0;
000001A8  3           INPE1 := INPE0;
000001AE  3
000001AE  3      END; ( ELDRDELAY )
00000000  2
00000000  2
00000000  2      PROCEDURE LOCODELAY(INPLDO : PUINT; VAR OUTLDO: PUINT);
00000000  3
00000000  3      ( Delay in the feedback path is simulated
00000000  3        using a 50 ms first order delay of the form:
00000000  3
00000000  3                  OUTLDO    =      1
00000000  3                  ------      -------------
00000000  3                  INPLDO       1 + (50ms)S                 }
00000000  3
00000000  3    CONST      A = 16059;     { 0.9802 PU }
00000000  3               B = 162;       { 0.0099 PU }
00000000  3
00000000  3               ABSMAX = 16383;     { 1.0 PU }
00000000  3               ABSMIN = -16383;    { -1.0 PU }
000001B6  3
000001B6  3    BEGIN
000001B6  3         OUTLDO := PUMULT(A,OUTLD1) + PUMULT(B,INPLDO)
000001B6  3                   + PUMULT(B,INPLD1);
000001FA  3
000001FA  3         IF OUTLDO > ABSMAX THEN
00000208  3              OUTLDO := ABSMAX;
00000210  3         IF OUTLDO < ABSMIN THEN
0000021E  3              OUTLDO := ABSMIN;
00000226  3
00000226  3         OUTLD1 := OUTLDO;
0000022E  3         INPLD1 := INPLDO;
00000234  3
00000234  3    END; { LOCODELAY }
00000000  2
00000000  2
00000000  2
00000000  2    PROCEDURE ACDRIVE(INP,NM,NL,SLOP : PUINT; VAR OUT : PUINT);
00000000  3
00000000  3    { The generation of torque in an AC drive is simultated.}
00000000  3
00000000  3    CONST     K1 = 16383;          { 1.0 PU }
00000000  3              K2 = 492;            { 0.03 PU }
00000000  3
00000000  3              F1Base = 6554;       { 0.4 PU }
00000000  3              NetK = 3072;         { .1875 PU = K2/(F1Base*F1Base) }
00000000  3              Unity = 16384;       { 1.0 PU }

00000000  3              Min = 0;             { 0.0 PU }
00000000  3
00000000  3    VAR     { F2 : PUINT;  }
00000000  3            { NMLIM,} NLSET : PUINT;
00000002  3              NUM,DENOM : PUINT;
00000006  3              F1SQ,FCTR : PUINT;
0000023C  3
0000023C  3                                        {                       2 }
0000023C  3    BEGIN                               { calculate K2 * (f1/f1b)  }
0000023C  3         IF F1 > F1Base THEN
0000024B  3              BEGIN
0000024B  3                   F1SQ := PUMULT(F1,F1);
00000256  3                   FCTR := PUMULT(F1SQ,NetK);
00000268  3              END
```

```
251 0000026C  3          ELSE
252 0000026C  3              FCTR := K2;
253 00000272  3
254 00000272  3          NUM := F1 - NM;                  ( calculate TMOT )
255 0000027E  3          DENOM := FCTR;
256 00000284  3
257 00000284  3          IF NUM > DENOM THEN
258 0000028C  3              BEGIN
259 0000028C  3                  NUM := NUM - DENOM;
260 00000294  3                  OUT := Unity + PUDIV(NUM,DENOM);
261 000002AA  3              END
262 000002AE  3          ELSE
263 000002AE  3              OUT := PUDIV(NUM,DENOM);
264 000002C2  3
265 000002C2  3          F2 := PUMULT(FCTR,INP);            ( calculate f2 )
266 000002D4  3
267 000002D4  3          NLSET := NL + SLOP;
268 000002E0  3
269 000002E0  3          IF NM > NLSET THEN
270 000002EA  3              NMLIM := NLSET
271 000002F2  3          ELSE
272 000002F2  3              NMLIM := NM;
273 000002F8  3
274 000002F8  3          F1 := F2 + NMLIM;                 ( calculate new F1 )
275 00000304  3
276 00000304  3      END; ( ACDRIVE )
277 00000000  2
278 00000000  2
279 00000000  2
280 00000000  2      PROCEDURE MTRPLANT(INPMO : PUINT; VAR OUTMO : PUINT);
281 00000000  3
282 00000000  3
283 00000000  3      (                   OUTMO   =      1
284 00000000  3                          ------     ---------
285 00000000  3                          INPMO      (1s)S
286 00000000  3
287 00000000  3           NOTE : This procedure, due to the small time
288 00000000  3      constant and the nature of the high gain feedback
289 00000000  3      in the circuit which it is a part, is executed with
290 00000000  3      a 1 ms cycle time.  Therefore, this procedure is
291 00000000  3      global and is executed in the interrupt procedure
292 00000000  3      PTMIRQ6.
293 00000000  3           Also, due to the sensitivity of the integrator
294 00000000  3      input, 32 bit resolution is maintained internal to
295 00000000  3      this procedure using LPUMULT.                      )

296 00000000  3
297 00000000  3      CONST     A = 9;  ( =         )
298 00000000  3
299 00000000  3               ABSMAX = 1073676288; ( = 1.0 * LCNVRT**2)
300 00000000  3               ABSMIN = 0;          ( = 0.0 PU )
```

```
301 00000000  3
302 00000000  3              LCNVRT = 256;
303 0000030C  3
304 0000030C  3
305 0000030C  3     BEGIN     { MTRPLANT }
306 0000030C  3
307 0000030C  3         OUTM1 := OUTM1 + LPUMULT(A,INPM0) + LPUMULT(A,INPM1);
308 0000033C  3
309 0000033C  3         IF OUTM1 > ABSMAX THEN
310 00000346  3              OUTM1 := ABSMAX;
311 0000034E  3         IF OUTM1 < ABSMIN THEN
312 00000356  3              OUTM1 := ABSMIN;
313 0000035A  3
314 0000035A  3         OUTM0 := SHIFT(OUTM1,-16);
315 0000036C  3
316 0000036C  3         INPM1 := INPM0;
317 00000372  3
318 00000372  3     END; (MTRPLANT)
319 00000000  2
320 00000000  2
321 00000000  2
322 00000000  2     PROCEDURE MLOSSES ( INP : PUINT; VAR OUT : PUINT);
323 00000000  3
324 00000000  3     { Motor losses ranging from 0.005 to 0.01 PU }
325 00000000  3     { REVISED LOSSES CONSTANT = .01 PU }
326 00000000  3
327 00000000  3     CONST     M = 82;  { slope = 0.005 PU }
328 00000000  3               B = 82;  { Y intercept = 0.005 PU }
329 0000037A  3
330 0000037A  3     BEGIN     { MLOSSES }
331 0000037A  3          OUT := PUMULT(M,INP) + B;
332 00000392  3     END;
333 00000000  2
334 00000000  2
335 00000000  2
336 00000000  2     PROCEDURE RAMPCHAR(QI2,QO2 : PUINT; VAR QI,QO : PUINT);
337 00000000  3
338 00000000  3     CONST     Tr = 100; { Tb / Ts }
339 00000000  3               MaxPU = 16383;
340 00000000  3               MinPU = 0;
341 00000000  3               Scale = 2;   {  DEFAULT = 1 }
342 00000000  3
343 00000000  3     VAR       DeltaQI,DeltaQO : PUINT;
344 00000004  3               QXI,QXO : PUINT;
345 0000039A  3
346 0000039A  3     BEGIN
347 0000039A  3         IF RAMPSTART THEN
348 000003A2  3              BEGIN
349 000003A2  3                   DeltaQI := QI2 - QI;
350 000003B0  3                   INCI := DeltaQI / Tr;
351 000003BA  3
```

```
352 000003BA  3                     DeltaQO := QO2 - QO;
353 000003C8  3                     INCO := DeltaQO / Tr;
354 000003D2  3

355 000003D2  3                     RAMPSTART := FALSE;
356 000003D6  3                     RAMPRUN := TRUE;
357 000003DC  3                     STEPCOUNT := 0;
358 000003E0  3                     SMPLCNT := 1;
359 000003E6  3                END
360 000003EA  3           ELSE
361 000003EA  3                IF RAMPRUN THEN
362 000003F2  3                     IF SMPLCNT >= (Scale * KFCTR) THEN
363 00000404  3                          BEGIN
364 00000404  3                               SMPLCNT := 1;
365 0000040A  3
366 0000040A  3                               QXI := QI + INCI;
367 00000418  3                               IF QXI > MaxPU THEN
368 00000420  3                                    QI := MaxPU
369 0000042C  3                               ELSE
370 0000042C  3                                    IF QXI < MinPU THEN
371 00000434  3                                         QI := MinPU
372 0000043E  3                                    ELSE
373 0000043E  3                                         QI := QXI;
374 00000446  3
375 00000446  3                               QXO := QO + INCO;
376 00000454  3                               IF QXO > MaxPU THEN
377 0000045C  3                                    QO := MaxPU
378 00000468  3                               ELSE
379 00000468  3                                    IF QXO < MinPU THEN
380 00000470  3                                         QO := MinPU
381 0000047A  3                                    ELSE
382 0000047A  3                                         QO := QXO;
383 00000482  3
384 00000482  3                               STEPCOUNT := STEPCOUNT + 1;
385 00000486  3                               IF STEPCOUNT > Tr THEN
386 00000492  3                                    RAMPRUN := FALSE;
387 00000496  3                          END
388 0000049A  3                    ELSE
389 0000049A  3                         SMPLCNT := SMPLCNT + 1;
390 0000049E  3
391 0000049E  3
392 0000049E  3      END; ( RAMPCHAR )
393 00000000  2
394 00000000  2
395 00000000  2
396 00000000  2      PROCEDURE FCC(GSlo : INTEGER; GLin,GnLin : PUINT;
397 00000000  3                     VAR FCProf : LIST;
398 00000000  3                     Inp : PUINT; VAR Out : PUINT);
399 00000000  3
400 00000000  3      CONST     MaxPU = 16383;
401 00000000  3                MinPU = -16383;
```

```
402 00000000  3
403 00000000  3     VAR       InpIndex : INTEGER;
404 000004A6  3
405 000004A6  3     BEGIN
406 000004A6  3          IF Inp > MaxPU THEN
407 000004B2  3               Out := MaxPU
408 000004BE  3          ELSE
409 000004BE  3               IF Inp > GLin THEN
410 000004CA  3                    BEGIN
411 000004CA  3                         InpIndex := SHIFT(Inp,-6);
412 000004D0  3                         Out := FCProf [InpIndex];
413 000004E2  3                    END

414 000004E6  3               ELSE
415 000004E6  3                    IF Inp > GnLin THEN
416 000004F2  3                         Out := GSlo * Inp
417 00000502  3                    ELSE
418 00000502  3                         IF Inp > MinPU THEN
419 0000050E  3                              BEGIN
420 0000050E  3                                   InpIndex := Inp / 64;
421 00000518  3                                   Out := FCProf [InpIndex];
422 0000052A  3                              END
423 0000052E  3                         ELSE
424 0000052E  3                              Out := MinPU;
425 00000536  3     END; { FCC }
426 0000053E  2
427 0000053E  2
428 0000053E  2
429 0000053E  2  BEGIN    { MOTORS  }
430 0000053E  2
431 0000053E  2      LOCODELAY(NLOCO,NLOCOD);
432 00000554  2
433 00000554  2      ACDRIVE(TEREF,NMOT,NLOCO,SLOP,TMOT1);  { NLOCO <=> NLOCOD }
434 00000578  2
435 00000578  2      ELDRDELAY(TMOT1,TMOT);
436 00000590  2
437 00000590  2      MTRPLANT(NMREF,NMOT);
438 000005A8  2
439 000005A8  2      MLOSSES(NMOT,LMOT);
440 000005C0  2
441 000005C0  2      SLIP := NMOT - NLOCO;
442 000005CE  2
443 000005CE  2      RAMPCHAR(TRGTI,TRGTO,MPRI,MPRO);
444 000005EC  2
445 000005EC  2      XSLIP := PUMULT(MPRI,SLIP);   { 16383 <=> MPRI }
446 000005FE  2
447 000005FE  2      WITH CURVESET [PROFILENUM] DO
448 00000614  2          FCC(SLOPE,LIN,NLIN,TABLE,XSLIP,XFLOCO);
449 0000063C  2
450 0000063C  2      FINT := PUMULT(MPRO,XFLOCO);   { 16383 <=> MPRO }
451 0000064E  2
```

```
 452 0000064E  2      FLOCO := FINT  {+ PUMULT(4096,FINT)} ;  {OUT * 1.25}
 453 00000656  2
 454 00000656  2      NMREF := TMOT - LMOT - FLOCO;
 455 0000066A  2
 456 0000066A  2  END;     { MOTORS  }
 457 00000000  1
 458 00000000  1
 459 00000000  1  $GLOBPROC OFF$
 460 00000000  1
 461 00000000  1  .   { main code in main program module. }

End of compilation, number of errors=    0
```

Program TLOCO

Procedure WINDAGE (INP : PUINT ; VAR OUT : PUINT);
The net forces acting on the locomotive are calculated as a function of the locomotive speed.

Procedure TRAINPLANT (INPL0 : PUINT ; VAR OUTL0 : PUINT)
The inertia of the "train" is simulated with a variable integrator ranging from T = 10s to 90s where $H(s) = \frac{1}{Ts}$.

Procedure DGT data sets for
DGIB real time display

```
00000000  1  "68000"
00000000  1  $EXTENSIONS$
00000000  1
00000000  1  PROGRAM TLOCO;
00000000  1
00000000  1
00000000  1      {    TLOCO is an external module containing the procedure
00000000  1           TRAIN which in turn contains all the procedures
00000000  1           for the train/locomotive portion of the locomotive
00000000  1           simulation.
00000000  1                The EXTERNAL variables listed below represent
00000000  1           globals from the main program module which are used
00000000  1           within this module.
00000000  1
00000000  1                                                               }
00000000  1  CONST    N = 8;
00000000  1
00000000  1  TYPE     PUINT = SIGNED_16;
00000000  1           INTEGER = SIGNED_16;
00000000  1           BYTE = SIGNED_8;
00000000  1           EIGHT = UNSIGNED_8;
00000000  1           LONGINT = SIGNED_32;
00000000  1           STATUSBITS = SET OF 0..7;
00000000  1           PROFILE = ARRAY [0..255] OF PUINT;
00000000  1           TABLE = ARRAY [1..5] OF PROFILE;
00000000  1
00000000  1
00000000  1  $EXTVAR ON$
00000000  1
00000000  1  VAR      LOCOTIME : INTEGER;
00000000  1           OUTL1,INPL1 : PUINT;
00000000  1           NLREF : PUINT;
00000000  1           EXTFCTR : INTEGER;
00000000  1           SELECT : CHAR;
00000000  1           DATASET : ARRAY [1..N] OF EIGHT;
00000000  1
00000000  1  $EXTVAR OFF$
00000000  1
00000000  1
00000000  1
00000000  1  FUNCTION PUMULT(XER,XAND : PUINT): PUINT; EXTERNAL;
00000000  1
00000000  1  FUNCTION PUDIV(DVND,DVSR : PUINT): PUINT; EXTERNAL;
00000000  1
00000000  1  FUNCTION ADJ(SXTN : PUINT): EIGHT; EXTERNAL;
00000000  1
00000000  1
00000000  1
00000000  1  $GLOBPROC ON$
00000000  1
```

```
 51 00000000  1
 52 00000000  1  PROCEDURE TRAIN(TEST,FLOCO : PUINT; VAR NLOCO : PUINT);
 53 00000000  2
 54 00000000  2
 55 00000000  2  VAR      NLOSS : PUINT;
 56 00000000  2
 57 00000000  2
 58 00000000  2
 59 00000000  2     PROCEDURE WINDAGE(INP: PUINT; VAR OUT : PUINT);

 60 00000000  3
 61 00000000  3     (    Losses in net moving force FLOCO are
 62 00000000  3          calculated as a function of train speed
 63 00000000  3          NLOCO.                                    )
 64 00000000  3
 65 00000000  3     CONST     M = 3604;      ( 0.22 PU )
 66 00000000  3               B = 0;          ( 0.00 PU )
 67 00000004  3
 68 00000004  3     BEGIN
 69 00000004  3          OUT := PUMULT(M,INP) + B;
 70 0000001C  3        (  OUT := INP; )
 71 0000001C  3
 72 0000001C  3     END; ( WINDAGE )
 73 00000000  2
 74 00000000  2
 75 00000000  2
 76 00000000  2     PROCEDURE TRAINPLANT(INPLO : PUINT; VAR OUTLO : PUINT);
 77 00000000  3
 78 00000000  3     (    The inertia of the train is simulated using the
 79 00000000  3          integrator function below with T = 10-60 sec.  So
 80 00000000  3          that accuracy may be preserved, this the sample
 81 00000000  3          period is increased to 10 Ts.
 82 00000000  3
 83 00000000  3                         OUTLO     =      1
 84 00000000  3                         -----       ---------
 85 00000000  3                         INPLO        (60)S                 )
 86 00000000  3
 87 00000000  3     CONST     A = 21;  ( 0.0125 PU )
 88 00000000  3
 89 00000000  3               AbsMax = 15973;     ( 2 **14 - 1 - 2A )
 90 00000000  3               AbsMin = 0;          ( 0.0 PU )
 91 00000000  3               IntFctr = 1;       ( sample time scaling )
 92 00000000  3
 93 00000000  3     VAR       ThOp : PUINT;
 94 00000024  3
 95 00000024  3     BEGIN
 96 00000024  3          LOCOTIME := LOCOTIME + 1;
 97 00000028  3          IF LOCOTIME >= (EXTFCTR * IntFctr) THEN
 98 0000003A  3          BEGIN
 99 0000003A  3               LOCOTIME := 0;
100 0000003E  3
```

```
101 0000003E  3                 ThOp := OUTL1 + PUMULT(A,INPL0)
102 0000003E  3                          + PUMULT(A,INPL1);
103 0000006E  3
104 0000006E  3                 IF ThOp > AbsMax THEN
105 00000076  3                      OUTL0 := AbsMax
106 00000082  3                 ELSE
107 00000082  3                      IF ThOp < AbsMin THEN
108 0000008A  3                           OUTL0 := AbsMin
109 00000094  3                      ELSE
110 00000094  3                           OUTL0 := ThOp;
111 0000009C  3
112 0000009C  3                 OUTL1 := OUTL0;
113 000000A4  3                 INPL1 := INPL0;
114 000000AA  3
115 000000AA  3            END;
116 000000AA  3
117 000000AA  3      END; { TRAINPLANT }
118 00000000  2

119 00000000  2
120 00000000  2
121 00000000  2     PROCEDURE DGT;
122 000000B2  3
123 000000B2  3     BEGIN
124 000000B2  3
125 000000B2  3          DATASET [2] := ADJ(TEST);     { TEST <=> NLOSS }
126 000000C4  3          DATASET [3] := ADJ(NLREF);
127 000000D2  3          DATASET [4] := ADJ(NLOCO);
128 000000E6  3          DATASET [5] := ADJ(FLOCO);
129 000000F8  3
130 000000F8  3
131 000000F8  3     END; { DGT }
132 00000000  2
133 00000000  2
134 00000000  2
135 00000000  2 PROCEDURE DGIB;
136 00000100  3
137 00000100  3 BEGIN
138 00000100  3
139 00000100  3
140 00000100  3     DATASET [3] := ADJ(FLOCO);
141 00000112  3     DATASET [4] := ADJ(NLOCO);
142 00000126  3
143 00000126  3
144 00000126  3 END; { DGIB }
145 0000012E  2
146 0000012E  2
147 0000012E  2
148 0000012E  2
149 0000012E  2 BEGIN { TRAIN }
150 0000012E  2
```

```
 151 0000012E  2      TRAINPLANT(NLREF,NLOCO);
 152 00000146  2
 153 00000146  2      WINDAGE(NLOCO,WLOSS);
 154 0000015E  2
 155 0000015E  2      NLREF := FLOCO - WLOSS - TEST;
 156 0000016E  2
 157 0000016E  2      IF SELECT = 'T' THEN DGT;
 158 00000184  2      IF SELECT = 'I' THEN DGIB;
 159 0000019A  2
 160 0000019A  2  END; { TRAIN }
 161 00000000  1
 162 00000000  1  $GLOBPROC OFF$
 163 00000000  1
 164 00000000  1  .   { main code in main program module. }

End of compilation, number of errors=    0
```

Program IOLOCO

Procedure SIGAH;

CA2 line of unused part 1 of the "upper" parallel port is raised.

Procedure SIGAL;

CA2 line of unused part 1 of the "upper" parallel port is lowered.

Procedure SIGBH; (not active)

CB2 line of unused port 1 of the "upper" parallel port is raised.

Procedure SIGBL; (not active)

CB2 line of unused port 1 of the "upper" parallel port is lowered.

Procedure IOSETUP;

Parallel ports are configured as follows:

Lower Port 2 A	input
Lower Port 2 B	input
Lower Port 1 A	output, CA 2 low
Lower Port 1 B	input, CB2 low
Upper Port 2 A	output, int. CA1 enabled, CA2 high
Upper Port 2 B	originally configured for A/D input (no longer active)

Procedure IO_PUTCHAR (Ch : CHAR);

IO_PUTCHAR and IO_PUTSTRING insert a character and a string, respectively into an output buffer : BUFFER. Once this is completed, the transmitter interrupts are enabled which allows the buffer to be transmitted (see XMITINT).

Procedure IO_PUTSTRING (Var String : STRING)

(see IO_PUTCHAR). OutString [o] gives the length of the passed string.

Procedure KEYBDINT;

Keyboard input servicing routine with lowest priority (level 2). This routine is accessed when there has been an interrupt request by the terminal serial port *and* interrupt routine AC1AIRQ2 has determined it was a received character interrupt. The routine identifies valid command requests, acknowledges them by sending an appropriate message to the terminal, and assigns the appropriate value to the relevant variable.

Procedure XMITINT;

Characters stored in BUFFER are sent to the serial port by this output servicing routine. XMITINT is invoked when there has been a terminal serial port interrupt request *and* the transmitter is ready to send another character. When control character '$' is detected, a CR and LF is outputted, under control of the flag LINED.

Procedure AC1AIRQ2;

Interrupt routine of priority 2 (lowest used) invoked by an interrupt request from the terminal serial port. Routine calls KEYBDINT for a received character interrupt, and XMITINT for a transmitter ready interrupt.

```
00000000  1      "68000"
00000000  1      $EXTENSIONS$
00000000  1
00000000  1  PROGRAM IOLOCO;
00000000  1
00000000  1  (   IOLOCO is a separate module that contains all the I/O
00000000  1      procedures for the locomotive simulation.  Variables
00000000  1      associated with the physical memory aspects of input and
00000000  1      output are defined within this module.  The EXTERNAL
00000000  1      variables listed here are global variables defined in LOCO
00000000  1      but used in this module.                                  )
00000000  1
00000000  1  CONST    N = 8;         ( = no. of channels being monitored )
00000000  1           M = 100;       ( = no. of samples/ channel  )
00000000  1           L = 800;       ( = total no. of samples / sweep  )
00000000  1
00000000  1
00000000  1      ( memory map offsets for upper and lower parallel ports )
00000000  1
00000000  1           PIRA1 = 1;     ( PORT A of PIA 1       )
00000000  1           PORA1 = 1;
00000000  1           DDRA1 = 1;
00000000  1           CRA1 = 3;
00000000  1
00000000  1           PIRB1 = 5;     ( PORT B of PIA 1       )
00000000  1           PORB1 = 5;
00000000  1           DDRB1 = 5;
00000000  1           CRB1 = 7;
00000000  1
00000000  1           PIRA2 = 9;     ( PORT A of PIA 2       )
00000000  1           PORA2 = 9;
00000000  1           DDRA2 = 9;
00000000  1           CRA2 = 11;
00000000  1
00000000  1           PIRB2 = 13;    ( PORT B of PIA 2       )
00000000  1           PORB2 = 13;
00000000  1           DDRB2 = 13;
00000000  1           CRB2 = 15;
00000000  1
00000000  1      ( memory map offsets for analog input port )
00000000  1
00000000  1           CS = 1;           ( analog input channel select )
00000000  1           IH = 3;           ( upper data byte and status )
00000000  1           IL = 5;           ( lower data byte )
00000000  1
00000000  1      ( memory map offsets for analog output port )
00000000  1
00000000  1           DA1L = 1;           ( DAC1 output, LS 8 bits )
00000000  1           DA1H = 3;           ( DAC1 output, MS 4 bits )
00000000  1           DA2L = 5;           ( DAC2   "    LS 8  "   )
```

```
  51 00000000  1          DA2H = 7;          { DAC2   *    MS 4   *  }
  52 00000000  1          DA3L = 9;          { DAC3   *    LS 8   *  }
  53 00000000  1          DA3H = 11;         { DAC3   *    MS 4   *  }
  54 00000000  1          DA4L = 13;         { DAC4   *    LS 8   *  }
  55 00000000  1          DA4H = 15;         { DAC4   *    MS 4   *  }
  56 00000000  1
  57 00000000  1          BFFRLNGTH = 100;
  58 00000000  1
  59 00000000  1

  60 00000000  1 TYPE     PUINT = SIGNED_16;
  61 00000000  1          INTEGER = SIGNED_16;
  62 00000000  1          BYTE = SIGNED_8;
  63 00000000  1          EIGHT = UNSIGNED_8;
  64 00000000  1          LONGINT = SIGNED_32;
  65 00000000  1          BITS = SET OF 0..7;
  66 00000000  1          MEMIO = ARRAY [0..16] OF BYTE;
****WARNING ??                                     ^508
  67 00000000  1          MEMIOUS = ARRAY [0..16] OF EIGHT;
****WARNING ??                                       ^508
  68 00000000  1          MEMIOBITS = ARRAY [0..16] OF BITS;
****WARNING ??                                         ^508
  69 00000000  1
  70 00000000  1 VAR
  71 00000000  1         $EXTVAR ON$
  72 00000000  1
  73 00000000  1          IP,OP : INTEGER;
  74 00000000  1          SELECT : CHAR;
  75 00000000  1          JOIN : BOOLEAN;
  76 00000000  1          KFCTR : INTEGER;
  77 00000000  1          RAMPSTART : BOOLEAN;
  78 00000000  1          PINDEX : INTEGER;
  79 00000000  1          RAMPON : BOOLEAN;
  80 00000000  1          LINEFD : BOOLEAN;
  81 00000000  1          LFIN : BOOLEAN;
  82 00000000  1          EXTFCTR : INTEGER;
  83 00000000  1
  84 00000000  1          DUMREAD : BYTE;
  85 00000000  1          I : INTEGER;
  86 00000000  1          DATASET : ARRAY [1..N] OF EIGHT;
  87 00000000  1
  88 00000000  1          MESSAGE : STRING;
  89 00000000  1
  90 00000000  1          MP1,MP2 : EIGHT;
  91 00000000  1          SM1,SM2 : EIGHT;
  92 00000000  1          PSETCNT,SWCNT : INTEGER;
  93 00000000  1          CDBINPA,CDBINPB : EIGHT;
  94 00000000  1
  95 00000000  1          STRTFLG,NTCH1FLG : BOOLEAN;
  96 00000000  1          STRTHYS,DLED : BOOLEAN;
  97 00000000  1          INTSLT,EXTSLT : BOOLEAN;
```

```
 98 00000000  1           MOTORING,BRAKING : BOOLEAN;
 99 00000000  1           TROOP : BOOLEAN;
100 00000000  1           MSCIN : BOOLEAN;
101 00000000  1
102 00000000  1          $EXTVAR OFF$
103 00000000  1
104 00000000  1           BUFFER : ARRAY [1..BFFRLNGTH] OF CHAR;
105 00000064  1           PREFIX : BOOLEAN;
106 00000065  1
107 00000065  1          $ORG 0FE61A0H$
108 00000065  1           LPORT : MEMIO;      ( lower // port, signed )
109 00000065  1          $ORG 0FE61A0H$
110 00000065  1           LPORTUS : MEMIOUS;  ( lower // port, unsigned )
111 00000065  1          $ORG 0FE61A0H$
112 00000065  1           LPORTBITS : MEMIOBITS;   ( lower // port, bit-mapped )
113 00000065  1
114 00000065  1          $ORG 0FE61E0H$
115 00000065  1           UPORT : MEMIO;      ( upper // port, signed )

116 00000065  1          $ORG 0FE61E0H$
117 00000065  1           UPORTUS : MEMIOUS;  ( upper // port, unsigned )
118 00000065  1          $ORG 0FE61E0H$
119 00000065  1           UPORTBITS : MEMIOBITS;   ( upper // port, bit-mapped )
120 00000065  1
121 00000065  1          $ORG 0FE6200H$
122 00000065  1           ADCUS : MEMIOUS;    ( A to D port, unsigned )
123 00000065  1          $ORG 0FE6200H$
124 00000065  1           ADCBITS : MEMIOBITS;  ( A to D port, bit-mapped )
125 00000065  1
126 00000065  1          $ORG 0FE6400H$
127 00000065  1           DAC : MEMIOUS;      ( D to A port, unsigned )
128 00000065  1
129 00000065  1          $ORG 0FE8001H$
130 00000065  1           ACIASTC : BITS;     ( term. serial port, status-cntrl )
131 00000065  1          $ORG 0FE8003H$
132 00000065  1           ACIAIOB : BYTE;     (  "      "     "   signed data  )
133 00000065  1          $ORG 0FE8003H$
134 00000065  1           ACIAIOC : CHAR;     (  "      "     "   char. data   )
135 00000065  1
136 00000065  1
137 00000065  1          $END_ORG$
138 00000000  1
139 00000000  1
140 00000000  1  FUNCTION PUMULT(XER,XAND : PUINT) : PUINT; EXTERNAL;
141 00000000  1
142 00000000  1  FUNCTION ADJ(SIXTN : PUINT) : EIGHT; EXTERNAL;
143 00000000  1
144 00000000  1
145 00000000  1  $GLOBPROC ON$
146 00000000  1
147 00000000  1  PROCEDURE SIGAH;
```

```
148 00000004  2  BEGIN
149 00000004  2      UPORTBITS [CRA1] := BITS[2,3,4,5];
150 0000000C  2  END;
151 00000000  1
152 00000000  1  PROCEDURE SIGAL;
153 00000014  2  BEGIN
154 00000014  2      UPORTBITS [CRA1] := BITS[2,4,5];
155 0000001C  2  END;
156 00000000  1
157 00000000  1      PROCEDURE SIGBH;
158 00000024  2  BEGIN
159 00000024  2       UPORTBITS [CRB1] := BITS[2,3,4,5];
160 0000002C  2  END;
161 00000000  1
162 00000000  1  PROCEDURE SIGBL;
163 00000034  2  BEGIN
164 00000034  2       UPORTBITS [CRB1] := BITS[2,4,5];
165 0000003C  2  END;
166 00000000  1
167 00000000  1
168 00000000  1
169 00000000  1  PROCEDURE IOSETUP;
170 00000044  2
171 00000044  2  BEGIN
172 00000044  2
173 00000044  2      PREFIX := FALSE;         { 'Keyboard numbers' access. }
174 00000048  2

175 00000048  2
176 00000048  2  {   train line inputs from control box }
177 00000048  2
178 00000048  2      LPORTBITS [CRA2] := BITS [];       { access to DDRA2 }
179 0000004E  2      LPORTBITS [DDRA2] := BITS [];      { configured as input }
180 00000054  2      LPORTBITS [CRA2] := BITS [2];
181 0000005C  2
182 0000005C  2
183 0000005C  2  {   switch inputs from control box, diesel }
184 0000005C  2  {   status LED output to control box       }
185 0000005C  2
186 0000005C  2      LPORTBITS [CRB2] := BITS [];       { access to DDRB2 }
187 00000062  2      LPORTBITS [DDRB2] := BITS [];      { configured as input }
188 00000068  2      LPORTBITS [CRB2] := BITS [2];
189 00000070  2
190 00000070  2
191 00000070  2  {   train line and switch outputs to Micas }
192 00000070  2
193 00000070  2      LPORTBITS [CRA1] := BITS [];       { access to DDRA1 }
194 00000076  2      LPORTBITS [DDRA1] := BITS [0,1,2,3,4,5,6,7];  { output }
195 0000007E  2      LPORTBITS [CRA1] := BITS [2,4,5];  { CA2 low }
196 00000086  2
197 00000086  2
```

```
198 00000086  2  {   train line and switch inputs from Micas, }
199 00000086  2  {   BTO switch output to Micas                  }
200 00000086  2
201 00000086  2     LPORTBITS [CRB1] := BITS [];       { access to DDRB1 }
202 000000BC  2     LPORTBITS [DDRB1] := BITS [];      { configured as input }
203 00000092  2     LPORTBITS [CRB1] := BITS [2,4,5];  { CB2 low }
204 0000009A  2
205 0000009A  2
206 0000009A  2  {   interrupt driven handshaking and data output }
207 0000009A  2  {   to "display routine" resident on 6809 board }
208 0000009A  2
209 0000009A  2     UPORTBITS [CRA2] := BITS [];       { access to DDRA2 }
210 000000A0  2     UPORTBITS [DDRA2] := BITS [0,1,2,3,4,5,6,7];  { output }
211 000000A8  2     UPORTBITS [CRA2] := BITS [0,2,3,4,5];   { int.CA1 enbld
212 000000B0  2                                               CA2 out high  }
213 000000B0  2
214 000000B0  2  {   temporary control output and data input }
215 000000B0  2  {   for AtoD 2                              }
216 000000B0  2
217 000000B0  2     UPORTBITS [CRB2] := BITS [];       { access to DDRB2 }
218 000000B6  2     UPORTBITS [DDRB2] := BITS [];      { configured as input }
219 000000BC  2     UPORTBITS [CRB2] := BITS [2,4,5]; {int. CB1 enbld,CA2 outlow}
220 000000C4  2
221 000000C4  2
222 000000C4  2  {   main loop monitoring }
223 000000C4  2  {   I/O port available   }
224 000000C4  2
225 000000C4  2     UPORTBITS [CRA1] := BITS [2,4,5];      { CA2 out low }
226 000000CC  2
227 000000CC  2
228 000000CC  2  {   temporary control output and data input }
229 000000CC  2  {   for AtoD 1                              }
230 000000CC  2
231 000000CC  2     UPORTBITS [CRB1] := BITS []; { access to DDRB1 }
232 000000D2  2     UPORTBITS [DDRB1] := BITS []; { configured as input }
233 000000D8  2     UPORTBITS [CRB1] := BITS [2,4,5];      { CB2 out low }

234 000000E0  2
235 000000E0  2
236 000000E0  2
237 000000E0  2     ACIASTC := BITS[0,1];   {Master RESET }
238 000000E8  2     ACIASTC := BITS[0,2,4,7]; { enable rec. ints, Xmit ints off }
239 000000F0  2
240 000000F0  2     ADCBITS [CS] := BITS [3];   { select ch 0, start conversion }
241 000000F8  2
242 000000F8  2  END; { IOSETUP }
243 00000000  1
244 00000000  1
245 00000000  1
246 00000000  1  PROCEDURE IO_PUTCHAR(Ch : CHAR);
247 00000100  2
```

```
248 00000100  2
249 00000100  2  BEGIN
250 00000100  2
251 00000100  2     IF IP > BFFRLNGTH THEN
252 0000010C  2         IP := 1;
253 00000112  2     BUFFER [IP] := Ch;
254 00000122  2     IP := IP + 1;
255 00000126  2
256 00000126  2     ACIASTC := BITS[0,2,4,5,7];  ( enable xmit interrupts )
257 0000012E  2
258 0000012E  2  END; ( IO_PUTCHAR )
259 00000000  1
260 00000000  1
261 00000000  1
262 00000000  1  PROCEDURE IO_PUTSTRING(VAR OutString : STRING);
263 00000000  2
264 00000000  2  (   Each output message is stored in a FIFO queue
265 00000000  2      where it awaits output by the interrupt procedure
266 00000000  2      XMITINT.                                        )
267 00000000  2
268 00000000  2
269 00000000  2  VAR      ByteCnt : BYTE;
270 00000001  2           Index : BYTE;
271 00000136  2
272 00000136  2  BEGIN
273 00000136  2     ByteCnt := BYTE(OutString [0]);
274 0000013E  2
275 0000013E  2     FOR Index := 1 TO ByteCnt DO
276 00000154  2     BEGIN
277 00000154  2         IF IP > BFFRLNGTH THEN
278 00000160  2             IP := 1;
279 00000166  2         BUFFER [IP] := OutString [Index];
280 00000180  2         IP := IP + 1;
281 00000184  2     END;
282 00000196  2
283 00000196  2     ACIASTC := BITS[0,2,4,5,7];  ( enable xmit interrupts )
284 0000019E  2
285 0000019E  2  END; ( IO_PUTSTRING )
286 00000000  1
287 00000000  1
288 00000000  1
289 00000000  1  PROCEDURE KEYBDINT;
290 00000000  2
291 00000000  2  CONST    OFFSET = 30H;   ( Re : ASCII table )
292 00000000  2

293 00000000  2  VAR      CHRCTR : CHAR;
294 00000001  2           OutChr : CHAR;
295 000001A6  2
296 000001A6  2  BEGIN
297 000001A6  2     CHRCTR := ACIAIOC;
```

```
000001AE  2
000001AE  2                CASE CHRCTR OF
000001B8  2
000001B8  2      'A','B','C','D',
000001B8  2      'E','I','N','T',
000001B8  2      'Y','Z'           : BEGIN
000001B8  2                              SELECT := CHRCTR;
000001BE  2                              MESSAGE := '$Data Group ';
000001CA  2                              IO_PUTSTRING(MESSAGE);
000001D4  2                              IO_PUTCHAR(CHRCTR);
000001DE  2                              MESSAGE := '  selected.$';
000001EA  2                              IO_PUTSTRING(MESSAGE);
000001F4  2                         END;
000001F8  2
000001F8  2      '1','2','3','4','5',
000001F8  2      '6','7','8','9' : BEGIN
000001F8  2                              IF PREFIX THEN
00000200  2                                   BEGIN
00000200  2                                        EXTFCTR := ORD(CHRCTR) - OFFSET;
00000210  2                                        MESSAGE := '$Integrator Factor = ';
0000021C  2                                        PREFIX := FALSE;
00000220  2                                   END
00000224  2                              ELSE
00000224  2                                   BEGIN
00000224  2                                        KFCTR := ORD(CHRCTR) - OFFSET;
00000234  2                                        MESSAGE := '$Time Base Factor = ';
00000240  2                                   END;
00000240  2
00000240  2                              IO_PUTSTRING(MESSAGE);
0000024A  2                              IO_PUTCHAR(CHRCTR);
00000254  2                              MESSAGE := '$';
0000025A  2                              IO_PUTSTRING(MESSAGE);
00000264  2                         END;
00000268  2
00000268  2            '!','@','#' : BEGIN
00000268  2                                   CASE CHRCTR OF
00000272  2
00000272  2                              '!' : PINDEX := 1;
0000027C  2                              '@' : PINDEX := 2;
00000286  2                              '#' : PINDEX := 3;
00000290  2                              OTHERWISE;
00000290  2
00000290  2                                   END; ( CASE )
000002A8  2
000002A8  2                              OutChr := CHR(PINDEX + OFFSET);
000002B4  2                              MESSAGE := '$FCC Profile ';
000002C0  2                              IO_PUTSTRING(MESSAGE);
000002CA  2                              IO_PUTCHAR(OutChr);
000002D4  2                              MESSAGE := ' selected.$';
000002E0  2                              IO_PUTSTRING(MESSAGE);
000002EA  2                         END;
```

```
000002EE  2
000002EE  2             'F' : BEGIN
000002EE  2                        LFIN := NOT LFIN;

000002F4  2                        IF LFIN THEN
000002FC  2                              MESSAGE := '$Load factor IN.$'
0000030C  2                        ELSE
0000030C  2                              MESSAGE := '$Load factor OUT.$';
00000318  2                        IO_PUTSTRING(MESSAGE);
00000322  2                   END;
00000326  2
00000326  2             'L' : BEGIN
00000326  2                        JOIN := NOT JOIN;
0000032C  2                        IF JOIN THEN
00000334  2                           MESSAGE :=
00000334  2                                 '$Diesel loaded by train.$'
00000344  2                        ELSE
00000344  2                           MESSAGE :=
00000344  2                        '$Diesel loaded by potentiometer.$';
00000350  2                        IO_PUTSTRING(MESSAGE);
0000035A  2                   END;
0000035E  2
0000035E  2             'O' : BEGIN
0000035E  2                        MSCIN := NOT MSCIN;
00000364  2                        IF MSCIN THEN
0000036C  2                             MESSAGE :=
0000036C  2                             '$Max Slip Ctrl IN.$'
0000037C  2                        ELSE
0000037C  2                             MESSAGE :=
0000037C  2                             '$Max Slip Ctrl OUT.$';
00000388  2                        IO_PUTSTRING(MESSAGE);
00000392  2                   END;
00000396  2
00000396  2             'P' : PREFIX := TRUE;
000003A0  2
000003A0  2             'R' : RAMPON := TRUE;
000003AA  2
000003AA  2             'S' : BEGIN
000003AA  2                        RAMPSTART := TRUE;
000003B0  2                        MESSAGE := '$Begin transition.$';
000003BC  2                        IO_PUTSTRING(MESSAGE);
000003C6  2                   END;
000003CA  2
000003CA  2        OTHERWISE BEGIN
000003CA  2                        MESSAGE :=
000003CA  2                    '$Invalid Input--Please select again.$';
000003D6  2                        IO_PUTSTRING(MESSAGE);
000003E0  2                   END;
000003E0  2
000003E0  2        END; { CASE }
00000478  2
```

```
399 00000478  2  END; ( KEYBDINT )
400 00000000  1
401 00000000  1
402 00000000  1
403 00000000  1      PROCEDURE XMITINT;
404 00000000  2
405 00000000  2      CONST     CarriageReturn = 0DH;
406 00000000  2                LineFeed = 0AH;
407 00000480  2
408 00000480  2
409 00000480  2      BEGIN
410 00000480  2          IF NOT LINEFD THEN     ( new char from buffer )

411 0000048A  2              IF BUFFER [OP] = '$' THEN    ( output CR,LF )
412 000004A0  2                  BEGIN
413 000004A0  2                      ACIAIOB := CarriageReturn;
414 000004A8  2                      LINEFD := TRUE;
415 000004AE  2                  END
416 000004B2  2              ELSE ( output character )
417 000004B2  2                  BEGIN
418 000004B2  2                      ACIAIOC := BUFFER [OP];
419 000004C4  2                      OP := OP + 1;
420 000004C8  2                      IF OP = IP THEN   ( disable Xmit ints. )
421 000004D4  2                          ACIASTC := BITS[0,2,4,7];
422 000004DC  2                      IF OP > BFFRLNGTH THEN   ( return )
423 000004E8  2                          OP := 1;
424 000004EE  2                  END
425 000004F2  2          ELSE  ( output LineFeed )
426 000004F2  2              BEGIN
427 000004F2  2                  ACIAIOB := LineFeed;
428 000004FA  2                  LINEFD := FALSE;
429 000004FE  2                  OP := OP + 1;
430 00000502  2                  IF OP = IP THEN
431 0000050E  2                          ACIASTC := BITS[0,2,4,7];
432 00000516  2                  IF OP > BFFRLNGTH THEN
433 00000522  2                      OP := 1;
434 00000528  2
435 00000528  2              END;
436 00000528  2
437 00000528  2       END; ( XMITINT )
438 00000000  1
439 00000000  1
440 00000000  1
441 00000000  1  $INTERRUPT ON$
442 00000000  1
443 00000000  1  PROCEDURE ACIAIRQ2;
444 00000000  2
445 00000000  2  (     interrupt routines for:
446 00000000  2           1) keyboard interrupts for "data group control"
447 00000000  2           2) transmit buffer interrupts for outputting
448 00000000  2              messages to the terminal.
```

```
00000000  2                                                          }
00000000  2  CONST    XmitFlag = 1;
00000000  2           RcvFlag = 0;
00000534  2
00000534  2
00000534  2  BEGIN
00000534  2
00000534  2     IF RcvFlag IN ACIASTC THEN
00000540  2          KEYBDINT
00000548  2     ELSE
00000548  2          XMITINT;
0000054C  2
0000054C  2  END; { ACIAIRQ2 }
00000000  1
00000000  1
00000000  1
00000000  1  PROCEDURE IRQ3;
0000055C  2
0000055C  2     { Interrupt procedure outputs member of DS   }
0000055C  2
0000055C  2

0000055C  2  BEGIN
0000055C  2       DUMREAD := UPORT [PORA2];      { clear interrupt }
00000564  2       IF I > N THEN I := 1;
00000576  2       UPORTUS [PORA2] := DATASET [I];       { send databyte to  6809 }
00000588  2       UPORT [CRA2] := 35H;
00000590  2       I := I + 1;
00000594  2       UPORT [CRA2] := 3DH;            { data available to 6809 }
0000059C  2  END; { IRQ3 }
00000000  1
00000000  1  $INTERRUPT OFF$
00000000  1
00000000  1
00000000  1
00000000  1  PROCEDURE IODATA(VAR PSET,NSET,TDSET,PESET,PTMTR,PDLIM : PUINT;
00000000  2                   VAR TRGTI,TRGTO,EXTST,NLOCOSET : PUINT;
00000000  2                   NMOT,NDIESEL,LDFCTR,NLOCO : PUINT);
00000000  2
00000000  2
00000000  2    { PSETCODE is a 4 bit code from the control box.  It is
00000000  2      converted to a Micas code which represents a
00000000  2      defined power limit-- PSET and a defined speed--NSET
00000000  2        These values are defined
00000000  2      within the CASE STATEMENT as is TDSET which is
00000000  2      derived from PSET and NSET.
00000000  2        TDSET  represents the torque necessary from the
00000000  2      diesel, allowing for auxillary power.
00000000  2        Additionally, PESET is a defined "traction
00000000  2      effort" power request.
00000000  2        PTMTR is an 8 bit value representing the
```

```
499 00000000  2        input potentiometer. STRTFLG and NTCH1FLG are
500 00000000  2        also set according to the input status.
501 00000000  2           MCODE is the binary representation that is
502 00000000  2        input to the MICAS.
503 00000000  2
504 00000000  2      All signals are represented as per unit values.
505 00000000  2
506 00000000  2      Defined diesel signals :
507 00000000  2
508 00000000  2      PSET_base = 3666.67 kW = 1.0 PU = 16383
509 00000000  2      NSET_base = 1100 RPM = 1.0 PU = 16383
510 00000000  2      TDSET_base = 31831.0 N.m = 1.0 PU = 16383
511 00000000  2
512 00000000  2      Defined motor signals :
513 00000000  2
514 00000000  2      PMOT_base = 14666.67 kW = 1.0 PU = 16383
515 00000000  2                = 4 * PSET_base
516 00000000  2      NMOT_base = 3300 RPM = 1.0 PU = 16383
517 00000000  2      TMOT_base = 42441 N.m = 1.0 PU = 16383
518 00000000  2
519 00000000  2      PESET:  100% PE
520 00000000  2                                                   }
521 00000000  2
522 00000000  2  CONST    EOC = 7;   { CA1 int. flag of port A2 }
523 00000000  2           OnePU = UNSIGNED_16(16383);
524 00000000  2           NegSignBit = UNSIGNED_16(8000H);
525 00000000  2
526 00000000  2  VAR      CPSETCODE,CSWCODE : BITS;
527 00000002  2           CPSETNUM,CSWNUM : EIGHT;
528 00000004  2           MCODE,MOUTCODE : BITS;

529 00000006  2           CDBSW : BITS;
530 00000007  2           MICASINP,MSWCODE : BITS;
531 00000009  2           MNSETCODE : BITS;
532 0000000A  2           MNSETUS : EIGHT;
533 0000000B  2           MNSETNUM : INTEGER;
534 0000000E  2           DAL,DAH : EIGHT;
535 00000010  2           INTSTP : UNSIGNED_16;
536 00000012  2           ATODIN1,ATODIN2 : PUINT;
537 00000016  2
538 00000016  2           SHOW : PUINT;
539 00000000  2
540 00000000  2
541 00000000  2      PROCEDURE DISPLINES(CODE : BITS);
542 00000000  3
543 00000000  3      {    Each of the four code lines from the control
544 00000000  3           box are displayed after they are debounced.      }
545 00000000  3
546 00000000  3
547 00000000  3      VAR       B0,B1,B2,B3 : PUINT;
548 000005A8  3
```

```
 549 000005A8  3        BEGIN
 550 000005A8  3
 551 000005A8  3            IF 0 IN CODE THEN
 552 000005B2  3                 B0 := 3277
 553 000005BC  3            ELSE
 554 000005BC  3                 B0 := 0;
 555 000005C0  3
 556 000005C0  3            IF 1 IN CODE THEN
 557 000005CA  3                 B1 := 6554
 558 000005D4  3            ELSE
 559 000005D4  3                 B1 := 3528;
 560 000005DA  3
 561 000005DA  3            IF 2 IN CODE THEN
 562 000005E4  3                 B2 := 9830
 563 000005EE  3            ELSE
 564 000005EE  3                 B2 := 6803;
 565 000005F4  3
 566 000005F4  3            IF 3 IN CODE THEN
 567 000005FE  3                 B3 := 13107
 568 00000608  3            ELSE
 569 00000608  3                 B3 := 10079;
 570 0000060E  3
 571 0000060E  3
 572 0000060E  3       {    IF SELECT = 'Z' THEN
 573 0000060E  3            BEGIN
 574 0000060E  3                 DATASET [1] := ADJ(B0);
 575 0000060E  3                 DATASET [2] := ADJ(B1);
 576 0000060E  3                 DATASET [3] := ADJ(B2);
 577 0000060E  3                 DATASET [4] := ADJ(B3);
 578 0000060E  3            END;                            }
 579 0000060E  3
 580 0000060E  3       END;
 581 00000000  2
 582 00000000  2
 583 00000000  2
 584 00000000  2       PROCEDURE DEBOUNCE(INP0 : EIGHT; VAR INP1,INP2 : EIGHT;
 585 00000000  3                      VAR COUNT : INTEGER; VAR PRESVALUE : EIGHT);
 586 00000616  3
 587 00000616  3       {    PRESVALUE maintains the last valid value of the

 588 00000616  3            input byte that is being debounced.  Input is sampled
 589 00000616  3            every 10 * 25 ms and PRESVALUE is updated only when
 590 00000616  3            three consecutive samples are equal.              }
 591 00000616  3
 592 00000616  3       BEGIN
 593 00000616  3
 594 00000616  3            COUNT := COUNT + 1;
 595 00000620  3
 596 00000620  3            IF COUNT = 10 THEN   { 250 msec interval ? }
 597 0000062E  3                 BEGIN
 598 0000062E  3                      COUNT := 0;
```

```
599 00000634  3                 IF (INPO = INP1) AND (INP1 = INP2) THEN
600 0000065C  3                     PRESVALUE := INPO;
601 00000662  3             END;
602 00000662  3
603 00000662  3         INP2 := INP1;
604 0000066C  3         INP1 := INPO;
605 00000674  3
606 00000674  3     END; { DEBOUNCE}
607 00000000  2
608 00000000  2
609 00000000  2
610 00000000  2     PROCEDURE ADINP(VAR OUT : PUINT);
611 00000000  3
612 00000000  3     {   This procedure converts the 8 lower bits and
613 00000000  3         4 upper bits of the 12 bit  input word from the
614 00000000  3         ADC into a 16 bit signed per unit integer.        }
615 00000000  3
616 00000000  3     CONST    InpAtoDH = UNSIGNED_16(2048);
617 00000000  3              InpAtoDL = EIGHT(8);
618 00000000  3              SignBit = 3;
619 00000000  3
620 00000000  3     VAR      UBITS : BITS;
621 00000001  3              USV : EIGHT;
622 0000067C  3
623 0000067C  3     BEGIN
624 0000067C  3
625 0000067C  3         UBITS := ADCBITS [IH] * BITS [0,1,2,3];  { mask upper 4 bits }
626 0000068A  3
627 0000068A  3         IF SignBit IN ADCBITS [IH] THEN  { convert u4b to 2's compl }
628 00000696  3              UBITS := UBITS * BITS [0,1,2]
629 000006A0  3         ELSE
630 000006A0  3              UBITS := UBITS + BITS [SignBit,4,5,6,7];
631 000006A6  3
632 000006A6  3         USV := EIGHT(UBITS);  { convert to an usigned number }
633 000006AC  3
634 000006AC  3         OUT := INTEGER(USV * InpAtoDH + ADCUS [IL] * InpAtoDL);
635 000006CA  3
636 000006CA  3  END; { ADINP }
637 00000000  2
638 00000000  2
639 00000000  2
640 00000000  2     PROCEDURE DAOUT(INP : PUINT; VAR OUTL,OUTH : EIGHT);
641 00000000  3
642 00000000  3     {   This procedure converts a signed 16 bit P.U. integer
643 00000000  3         to a low byte and upper byte of a 12 bit binary
644 00000000  3         offset code format used by the DAC.              }
645 00000000  3
646 00000000  3     CONST    DtoAAdjL = EIGHT(8);

647 00000000  3              DtoAAdjH = EIGHT(256);
648 00000000  3              CNVRT = EIGHT(08H);    { applied to upper 4 bits }
```

```
 649 00000000  3
 650 00000000  3       VAR       OUTW : UNSIGNED_16;
 651 000006D2  3
 652 000006D2  3       BEGIN
 653 000006D2  3
 654 000006D2  3           OUTW := UNSIGNED_16(INP) / DtoAAdjL;  { shift to 12 bit }
 655 000006E0  3           OUTL := EIGHT(OUTW);     { lower byte }
 656 000006E6  3           OUTH := EIGHT(OUTW / DtoAAdjH);
 657 000006F6  3
 658 000006F6  3           IF OUTH >= CNVRT THEN   { convert to binary offset }
 659 00000704  3               OUTH := OUTH - CNVRT
 660 00000712  3           ELSE
 661 00000712  3               OUTH := OUTH + CNVRT;
 662 0000071C  3
 663 0000071C  3       END; { DAOUT }
 664 00000724  2
 665 00000724  2
 666 00000724  2
 667 00000724  2   BEGIN
 668 00000724  2
 669 00000724  2      { input and debounce control box lines }
 670 00000724  2
 671 00000724  2      CPSETCODE := LPORTBITS [PIRA2];      { get tr lines code }
 672 0000072C  2      CPSETCODE := CPSETCODE * BITS [0,1,2,3];   { mask upper 4 bits }
 673 00000732  2      CPSETNUM := EIGHT(CPSETCODE);    { convert to an unsigned num. }
 674 00000738  2
 675 00000738  2      DEBOUNCE(CPSETNUM,MP1,MP2,PSETCNT,CDBINPA);
 676 0000075A  2
 677 0000075A  2      CSWCODE := LPORTBITS [PIRB2];        { get switch codes }
 678 00000762  2      CSWCODE := CSWCODE * BITS [0,1,2,3,4,5,7];    { mask unused bits }
 679 00000768  2      CSWNUM := EIGHT(CSWCODE);        { convert to an unsigned num. }
 680 0000076E  2
 681 0000076E  2      DEBOUNCE(CSWNUM,SM1,SM2,SWCNT,CDBINPB);
 682 00000790  2
 683 00000790  2
 684 00000790  2      DISPLINES( BITS(CDBINPA) );   { display tr lines }
 685 0000079E  2
 686 0000079E  2
 687 0000079E  2
 688 0000079E  2      { convert applicable signals to Micas format
 689 0000079E  2        and output to Micas                        }
 690 0000079E  2
 691 0000079E  2               CASE INTEGER(CDBINPA) OF
 692 000007AC  2
 693 000007AC  2          0FH: MCODE := BITS [0,3];     { notch 0 }
 694 000007B6  2
 695 000007B6  2          07H: MCODE := BITS [];        { notch 1 }
 696 000007BE  2
 697 000007BE  2          0BH: MCODE := BITS [3];       { notch 2 }
 698 000007C8  2
 699 000007C8  2          03H: MCODE := BITS [1];       { notch 3 }
```

```
700 000007D2  2
701 000007D2  2            0DH: MCODE := BITS [1,3];     { notch 4 }
702 000007DC  2
703 000007DC  2            05H: MCODE := BITS [0,1,2];   { notch 5 }
704 000007E6  2
705 000007E6  2            09H: MCODE := BITS [0,1,2,3]; { notch 6 }

706 000007F0  2
707 000007F0  2            01H: MCODE := BITS [1,2];     { notch 7 }
708 000007FA  2
709 000007FA  2            0EH: MCODE := BITS [1,2,3];   { notch 8 }
710 00000804  2
711 00000804  2            06H: MCODE := BITS [0,3];       { notch 9 }
712 0000080E  2
713 0000080E  2            OTHERWISE;  { no other possibilities }
714 0000080E  2
715 0000080E  2                 END; { CASE }
716 0000084C  2
717 0000084C  2       MOUTCODE := BITS [];              { clear old output code }
718 00000850  2       MOUTCODE := MOUTCODE + MCODE;     { add in new MCODE }
719 00000858  2
720 00000858  2
721 00000858  2       CDBSW := BITS(CDBINPB);
722 0000085E  2       STRTFLG := FALSE;
723 00000862  2
724 00000862  2       IF NOT (0 IN CDBSW) THEN     { Forward }
725 00000876  2            MOUTCODE := MOUTCODE + BITS [4];
726 0000087C  2
727 0000087C  2       IF NOT (1 IN CDBSW) THEN     { Reverse }
728 00000890  2            MOUTCODE := MOUTCODE + BITS [5];
729 00000896  2
730 00000896  2       IF 2 IN CDBSW THEN          { M/B }
731 000008A0  2                 LPORTBITS [CRA1] := BITS [2,3,4,5]
732 000008AC  2       ELSE
733 000008AC  2                 LPORTBITS [CRA1] := BITS [2,4,5];
734 000008B4  2
735 000008B4  2       IF NOT (3 IN CDBSW) THEN     { INTLT }
736 000008C8  2                 MOUTCODE := MOUTCODE + BITS [6];
737 000008CE  2
738 000008CE  2       IF NOT (4 IN CDBSW) THEN     { EXTLT }
739 000008E2  2                 MOUTCODE := MOUTCODE + BITS [7];
740 000008E8  2
741 000008E8  2       IF 5 IN CDBSW THEN          { BTO }
742 000008F2  2            LPORTBITS [CRB1] := BITS [2,3,4,5]
743 000008FE  2       ELSE
744 000008FE  2            LPORTBITS [CRB1] := BITS [2,4,5];
745 00000906  2
746 00000906  2       IF NOT (7 IN CDBSW) THEN     { START }
747 0000091A  2            STRTFLG := TRUE
748 00000924  2       ELSE
749 00000924  2            STRTHYS := FALSE;
```

```
750 00000928  2
751 00000928  2      LPORTBITS [PORA1] := MOUTCODE;     { send data to Micas }
752 00000930  2
753 00000930  2      IF DLED THEN   { control CB2 according to DLED }
754 00000938  2           LPORTBITS [CRB2] := BITS[2,3,4,5]
755 00000944  2      ELSE
756 00000944  2           LPORTBITS [CRB2] := BITS[2,4,5];
757 0000094C  2
758 0000094C  2
759 0000094C  2      { input Micas train line and switch codes and assign values
760 0000094C  2        from table.                                    }
761 0000094C  2
762 0000094C  2      MICASINP := LPORTBITS [PIRB1];
763 00000954  2
764 00000954  2      MNSETCODE := MICASINP * BITS [0,1,2,3];

765 00000960  2      MNSETUS := EIGHT(MNSETCODE);
766 00000964  2      MNSETNUM := INTEGER(MNSETUS);
767 0000096C  2
768 0000096C  2      MSWCODE := MICASINP * BITS [4,5,6,7];
769 00000978  2
770 00000978  2      NTCHIFLG := FALSE;       { initialize notch flag }
771 0000097C  2
772 0000097C  2
773 0000097C  2                 CASE MNSETNUM OF
774 00000984  2
775 00000984  2          09H: BEGIN     { notch 0 }
776 00000984  2                      PSET := 164;          { 0.0100 PU }
777 0000098C  2                      NSET := 5958;       { 0.3636 PU }
778 00000994  2                      TDSET := 3088;      { 0.1885 PU }
779 0000099C  2                      NTCHIFLG := TRUE;
780 000009A2  2                   {  TESET := 0;  }
781 000009A2  2                 END;
782 000009A6  2
783 000009A6  2          00H: BEGIN      { notch 1 }
784 000009A6  2                      PSET := 164;          { 0.0100 PU }
785 000009AE  2                      NSET := 5958;       { 0.3636 PU }
786 000009B6  2                      TDSET := 3088;      { 0.1885 PU }
787 000009BE  2                      NTCHIFLG := TRUE;
788 000009C4  2                   {  TESET := 2048; }
789 000009C4  2                 END;
790 000009C8  2
791 000009C8  2          08H: BEGIN      { notch 2 }
792 000009C8  2                      PSET := 460;          { 0.0281 PU }
793 000009D0  2                      NSET := 7298;       { 0.4455 PU }
794 000009D8  2                      TDSET := 3775;      { 0.2304 PU }
795 000009E0  2                   {  TESET := 4096; }
796 000009E0  2                 END;
797 000009E4  2
798 000009E4  2          02H: BEGIN      { notch 3 }
799 000009E4  2                      PSET := 1055;        { 0.0644 PU }
```

```
800 000009EC  2                     NSET := 8639;      { 0.5273 PU }
801 000009F4  2                     TDSET := 4461;     { 0.2723 PU }
802 000009FC  2                   { TESET := 6144; }
803 000009FC  2                 END;
804 00000A00  2
805 00000A00  2             0AH: BEGIN     { notch 4 }
806 00000A00  2                     PSET := 1845;       { 0.1126 PU }
807 00000A08  2                     NSET := 10128;      { 0.6182 PU }
808 00000A10  2                     TDSET := 5147;     { 0.3142 PU }
809 00000A18  2                   { TESET := 8192; }
810 00000A18  2                 END;
811 00000A1C  2
812 00000A1C  2             07H: BEGIN     { notch 5 }
813 00000A1C  2                     PSET := 3418;       { 0.2086 PU }
814 00000A24  2                     NSET := 11469;      { 0.7000 PU }
815 00000A2C  2                     TDSET := 7437;     { 0.4539 PU }
816 00000A34  2                   { TESET := 10240 }
817 00000A34  2                 END;
818 00000A38  2
819 00000A38  2             0FH: BEGIN     { notch 6 }
820 00000A38  2                     PSET := 5912;      { 0.3608 PU }
821 00000A40  2                     NSET := 12809;      { 0.7818 PU }
822 00000A48  2                     TDSET := 9848;     { 0.6011 PU }
823 00000A50  2                   { TESET := 12288; }

824 00000A50  2                 END;
825 00000A54  2
826 00000A54  2             06H: BEGIN     { notch 7 }
827 00000A54  2                     PSET := 8865;      { 0.5411 PU }
828 00000A5C  2                     NSET := 14299;      { 0.8727 PU }
829 00000A64  2                     TDSET := 12206;     { 0.7450 PU }
830 00000A6C  2                   { TESET := 14336; }
831 00000A6C  2                 END;
832 00000A70  2
833 00000A70  2             0EH: BEGIN     { notch 8 }
834 00000A70  2                     PSET := 13334;      { 0.8138 PU }
835 00000A78  2                     NSET := 15639;      { 0.9546 PU }
836 00000A80  2                     TDSET := 15841;     { 0.9669 PU }
837 00000A88  2                   { TESET := 16383; }
838 00000A88  2                 END;
839 00000A8C  2
840 00000A8C  2             01H: ;         { notch 9 }
841 00000A90  2
842 00000A90  2
843 00000A90  2             OTHERWISE ;    { should be no other possibilities }
844 00000A90  2
845 00000A90  2                 END; { CASE }
846 00000ACE  2
847 00000ACE  2
848 00000ACE  2       IF 4 IN MSWCODE THEN     { BBR1 }
849 00000AD8  2           BRAKING := TRUE
```

```
 850 00000AE2  2       ELSE
 851 00000AE2  2            BRAKING := FALSE;
 852 00000AE6  2
 853 00000AE6  2       IF 5 IN MSWCODE THEN     { BFA1 }
 854 00000AF0  2            MOTORING := TRUE
 855 00000AFA  2       ELSE
 856 00000AFA  2            MOTORING := FALSE;
 857 00000AFE  2
 858 00000AFE  2       IF 6 IN MSWCODE THEN     { BSLT2 }
 859 00000B08  2            EXTSLT := TRUE
 860 00000B12  2       ELSE
 861 00000B12  2            EXTSLT := FALSE;
 862 00000B16  2
 863 00000B16  2       IF 7 IN MSWCODE THEN     { BSLT1 }
 864 00000B20  2            INTSLT := TRUE
 865 00000B2A  2       ELSE
 866 00000B2A  2            INTSLT := FALSE;
 867 00000B2E  2
 868 00000B2E  2
 869 00000B2E  2       IF TROOP THEN            { torque/power signal }
 870 00000B36  2        BEGIN
 871 00000B36  2           SIGBH;
 872 00000B3A  2           SHOW := 16000;
 873 00000B40  2        END
 874 00000B44  2       ELSE
 875 00000B44  2        BEGIN
 876 00000B44  2           SIGBL;
 877 00000B48  2           SHOW := 0;
 878 00000B4C  2        END;
 879 00000B4C  2
 880 00000B4C  2       { input from Analog to Digital port }
 881 00000B4C  2
 882 00000B4C  2

 883 00000B4C  2       REPEAT    { ensure AtoD conversion (ch 0) complete }
 884 00000B4C  2       UNTIL EOC IN ADCBITS [IH];
 885 00000B56  2
 886 00000B56  2       ADINP(PTMTR);
 887 00000B66  2
 888 00000B66  2
 889 00000B66  2       ADCBITS [CS] := BITS [0,3]; { select ch 1,start conversion }
 890 00000B6E  2
 891 00000B6E  2       REPEAT
 892 00000B6E  2       UNTIL EOC IN ADCBITS [IH];
 893 00000B78  2
 894 00000B78  2       ADINP(PESET);
 895 00000B88  2
 896 00000B88  2
 897 00000B88  2       ADCBITS [CS] := BITS [1,3];  { select ch 2, start conv. }
 898 00000B90  2
 899 00000B90  2       REPEAT
```

```
900 00000B90  2       UNTIL EOC IN ADCBITS [IH];
901 00000B9A  2
902 00000B9A  2       ADINP(PDLIM);
903 00000BAA  2
904 00000BAA  2       ADCBITS [CS] := BITS [0,1,3]; { select ch 3, start conv. }
905 00000BB2  2
906 00000BB2  2       REPEAT
907 00000BB2  2       UNTIL EOC IN ADCBITS [IH];
908 00000BBC  2
909 00000BBC  2       ADINP(TRGTI);
910 00000BCC  2
911 00000BCC  2       ADCBITS [CS] := BITS [2,3]; { select ch 4, start conv. }
912 00000BD4  2
913 00000BD4  2       REPEAT
914 00000BD4  2       UNTIL EOC IN ADCBITS [IH];
915 00000BDE  2
916 00000BDE  2       ADINP(TRGTO);
917 00000BEE  2
918 00000BEE  2       ADCBITS [CS] := BITS [0,2,3]; { select ch 5, start conv. }
919 00000BF6  2
920 00000BF6  2       REPEAT
921 00000BF6  2       UNTIL EOC IN ADCBITS [IH];
922 00000C00  2
923 00000C00  2       ADINP(EXTST);
924 00000C10  2
925 00000C10  2       ADCBITS [CS] := BITS[1,2,3];  { select ch 6, start conv. }
926 00000C18  2
927 00000C18  2       REPEAT
928 00000C18  2       UNTIL EOC IN ADCBITS [IH];
929 00000C22  2
930 00000C22  2       ADINP(NLOCOSET);
931 00000C32  2
932 00000C32  2       ADCBITS [CS] := BITS [3];    { select ch 0, start conv. }
933 00000C3A  2
934 00000C3A  2
935 00000C3A  2       { output to DAC }
936 00000C3A  2
937 00000C3A  2       DAOUT(NMOT,DAL,DAH);
938 00000C54  2       DAC [DA1L] := DAL;
939 00000C5C  2       DAC [DA1H] := DAH;
940 00000C64  2
941 00000C64  2       DAOUT(NDIESEL,DAL,DAH);

942 00000C7E  2       DAC [DA2L] := DAL;
943 00000C86  2       DAC [DA2H] := DAH;
944 00000C8E  2
945 00000C8E  2       DAOUT(NLOCO,DAL,DAH);
946 00000CA8  2       DAC [DA3L] := DAL;
947 00000CB0  2       DAC [DA3H] := DAH;
948 00000CB8  2
949 00000CB8  2       DAOUT(0,DAL,DAH);  {0 <=> LDFCTR}
```

```
950 00000CD0  2      DAC [DA4L] := DAL;
951 00000CD8  2      DAC [DA4H] := DAH;
952 00000CE0  2
953 00000CE0  2
954 00000CE0  2      IF SELECT = 'Y' THEN
955 00000CEC  2      BEGIN
956 00000CEC  2           DATASET [1] := ADJ(PTMTR);
957 00000CFC  2           DATASET [2] := ADJ(PESET);
958 00000D0C  2           DATASET [3] := ADJ(PDLIM);
959 00000D1C  2           DATASET [4] := ADJ(TRGTI);
960 00000D2C  2           DATASET [5] := ADJ(TRGTO);
961 00000D3C  2           DATASET [6] := ADJ(EXTST);
962 00000D4C  2           DATASET [7] := ADJ(NDIESEL);
963 00000D5A  2      END;
964 00000D5A  2
965 00000D5A  2      IF SELECT = 'Z' THEN
966 00000D66  2      BEGIN
967 00000D66  2           DATASET [1] := ADJ(LDFCTR);
968 00000D74  2           DATASET [2] := ADJ(PESET);
969 00000D84  2           DATASET [3] := ADJ(NLOCOSET);
970 00000D94  2           DATASET [4] := ADJ(NLOCO);
971 00000DA2  2           DATASET [5] := ADJ(SHOW);
972 00000DB0  2           DATASET [6] := ADJ(NMOT);
973 00000DBE  2           DATASET [7] := ADJ(NDIESEL);
974 00000DCC  2      END;
975 00000DCC  2
976 00000DCC  2  END; { IODATA }
977 00000000  1
978 00000000  1
979 00000000  1
980 00000000  1  $GLOBPROC OFF$
981 00000000  1  .
982 00000000  1

508: Warning: field or entry alignment; record or array comparisons may not work

End of compilation, number of errors=    0
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                        1          "68000"
                        2          NAME "DATA FOR SLIP CURVE(S)."
                        3          DATA
                        4          GLOBAL   CURVESET
                        5
 000000                 6 CURVESET
 000000 0032 00A4       7          DC.W              50,         164,        -164
                        8
 000006 C17C C198       9          DC.W      -16004, -15976, -15947, -15918, -15889
 000010 C20C C228      10          DC.W      -15860, -15832, -15803, -15774, -15745
 00001A C29C C2B8      11          DC.W      -15716, -15688, -15659, -15630, -15601
 000024 C32C C348      12          DC.W      -15572, -15544, -15515, -15486, -15457
 00002E C3BC C3D8      13          DC.W      -15428, -15400, -15371, -15342, -15313
 000038 C44C C468      14          DC.W      -15284, -15256, -15227, -15198, -15169
 000042 C4DC C4F8      15          DC.W      -15140, -15112, -15083, -15054, -15025
 00004C C56C C588      16          DC.W      -14996, -14968, -14939, -14910, -14881
 000056 C5FC C618      17          DC.W      -14852, -14824, -14795, -14766, -14737
 000060 C68C C6A8      18          DC.W      -14708, -14680, -14651, -14622, -14593
 00006A C71C C738      19          DC.W      -14564, -14536, -14507, -14478, -14449
 000074 C7AC C7C8      20          DC.W      -14420, -14392, -14363, -14334, -14305
 00007E C83C C858      21          DC.W      -14276, -14248, -14219, -14190, -14161
 000088 C8CC C8E8      22          DC.W      -14132, -14104, -14075, -14046, -14017
 000092 C95C C978      23          DC.W      -13988, -13960, -13931, -13902, -13873
 00009C C9EC CA08      24          DC.W      -13844, -13816, -13787, -13758, -13729
 0000A6 CA7C CA98      25          DC.W      -13700, -13672, -13643, -13614, -13585
 0000B0 CB0C CB28      26          DC.W      -13556, -13528, -13499, -13470, -13441
 0000BA CB9C CBB8      27          DC.W      -13412, -13384, -13355, -13326, -13297
 0000C4 CC2C CC48      28          DC.W      -13268, -13240, -13211, -13182, -13153
 0000CE CCBC CCD8      29          DC.W      -13124, -13096, -13067, -13038, -13009
 0000D8 CD4C CD68      30          DC.W      -12980, -12952, -12923, -12894, -12865
 0000E2 CDDC CDF8      31          DC.W      -12836, -12808, -12779, -12750, -12721
 0000EC CE6C CE88      32          DC.W      -12692, -12664, -12635, -12606, -12577
 0000F6 CEFC CF18      33          DC.W      -12548, -12520, -12491, -12462, -12433
 000100 CF8C CFA8      34          DC.W      -12404, -12376, -12347, -12318, -12289
 00010A D01C D038      35          DC.W      -12260, -12232, -12203, -12174, -12145
 000114 D0AC D0C8      36          DC.W      -12116, -12088, -12059, -12030, -12001
 00011E D13C D158      37          DC.W      -11972, -11944, -11915, -11886, -11857
 000128 D1CC D1E8      38          DC.W      -11828, -11800, -11771, -11742, -11713
 000132 D25C D278      39          DC.W      -11684, -11656, -11627, -11598, -11569
 00013C D2EC D308      40          DC.W      -11540, -11512, -11483, -11454, -11425
 000146 D37C D398      41          DC.W      -11396, -11368, -11339, -11310, -11281
 000150 D40C D428      42          DC.W      -11252, -11224, -11195, -11166, -11137
 00015A D49C D4B8      43          DC.W      -11108, -11080, -11051, -11022, -10993
 000164 D52C D548      44          DC.W      -10964, -10936, -10907, -10878, -10849
 00016E D5BC D5D8      45          DC.W      -10820, -10792, -10763, -10734, -10705
 000178 D64C D668      46          DC.W      -10676, -10648, -10619, -10590, -10561
 000182 D6DC D6F8      47          DC.W      -10532, -10504, -10475, -10446, -10417
 00018C D76C D788      48          DC.W      -10388, -10360, -10331, -10302, -10273
 000196 D7FC D818      49          DC.W      -10244, -10216, -10187, -10158, -10129
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

 0001A0 DBBC DBA8      50          DC.W       -10100, -10072, -10043, -10014, -9985
 0001AA D91C D938      51          DC.W       -9956, -9928, -9899, -9870, -9841
 0001B4 D9AC D9C8      52          DC.W       -9812, -9784, -9755, -9726, -9697
 0001BE DA3C DA58      53          DC.W       -9668, -9640, -9611, -9582, -9553
 0001C8 DACC DAE8      54          DC.W       -9524, -9496, -9467, -9438, -9409
 0001D2 DB5C DB78      55          DC.W       -9380, -9352, -9323, -9294, -9265
 0001DC DBEC DC08      56          DC.W       -9236, -9208, -9179, -9150, -9121
 0001E6 DC7C DC98      57          DC.W       -9092, -9064, -9035, -9006, -8977


 0001F0 DD0C DD28      58          DC.W       -8948, -8920, -8887, -8846, -8796
 0001FA DDE1 DE2F      59          DC.W       -8735, -8657, -8553, -8389,     0
 000204 0000 0000      60          DC.W           0,     0,  8389,  8553,  8657
 00020E 221F 225C      61          DC.W        8735,  8796,  8846,  8887,  8920
 000218 22F4 2311      62          DC.W        8948,  8977,  9006,  9035,  9064
 000222 2384 23A1      63          DC.W        9092,  9121,  9150,  9179,  9208
 00022C 2414 2431      64          DC.W        9236,  9265,  9294,  9323,  9352
 000236 24A4 24C1      65          DC.W        9380,  9409,  9438,  9467,  9496
 000240 2534 2551      66          DC.W        9524,  9553,  9582,  9611,  9640
 00024A 25C4 25E1      67          DC.W        9668,  9697,  9726,  9755,  9784
 000254 2654 2671      68          DC.W        9812,  9841,  9870,  9899,  9928
 00025E 26E4 2701      69          DC.W        9956,  9985, 10014, 10043, 10072
 000268 2774 2791      70          DC.W       10100, 10129, 10158, 10187, 10216
 000272 2804 2821      71          DC.W       10244, 10273, 10302, 10331, 10360
 00027C 2894 28B1      72          DC.W       10388, 10417, 10446, 10475, 10504
 000286 2924 2941      73          DC.W       10532, 10561, 10590, 10619, 10648
 000290 29B4 29D1      74          DC.W       10676, 10705, 10734, 10763, 10792
 00029A 2A44 2A61      75          DC.W       10820, 10849, 10878, 10907, 10936
 0002A4 2AD4 2AF1      76          DC.W       10964, 10993, 11022, 11051, 11080
 0002AE 2B64 2B81      77          DC.W       11108, 11137, 11166, 11195, 11224
 0002B8 2BF4 2C11      78          DC.W       11252, 11281, 11310, 11339, 11368
 0002C2 2C84 2CA1      79          DC.W       11396, 11425, 11454, 11483, 11512
 0002CC 2D14 2D31      80          DC.W       11540, 11569, 11598, 11627, 11656
 0002D6 2DA4 2DC1      81          DC.W       11684, 11713, 11742, 11771, 11800
 0002E0 2E34 2E51      82          DC.W       11828, 11857, 11886, 11915, 11944
 0002EA 2EC4 2EE1      83          DC.W       11972, 12001, 12030, 12059, 12088
 0002F4 2F54 2F71      84          DC.W       12116, 12145, 12174, 12203, 12232
 0002FE 2FE4 3001      85          DC.W       12260, 12289, 12318, 12347, 12376
 000308 3074 3091      86          DC.W       12404, 12433, 12462, 12491, 12520
 000312 3104 3121      87          DC.W       12548, 12577, 12606, 12635, 12664
 00031C 3194 31B1      88          DC.W       12692, 12721, 12750, 12779, 12808
 000326 3224 3241      89          DC.W       12836, 12865, 12894, 12923, 12952
 000330 32B4 32D1      90          DC.W       12980, 13009, 13038, 13067, 13096
 00033A 3344 3361      91          DC.W       13124, 13153, 13182, 13211, 13240
 000344 33D4 33F1      92          DC.W       13268, 13297, 13326, 13355, 13384
 00034E 3464 3481      93          DC.W       13412, 13441, 13470, 13499, 13528
 000358 34F4 3511      94          DC.W       13556, 13585, 13614, 13643, 13672
 000362 3584 35A1      95          DC.W       13700, 13729, 13758, 13787, 13816
 00036C 3614 3631      96          DC.W       13844, 13873, 13902, 13931, 13960
 000376 36A4 36C1      97          DC.W       13988, 14017, 14046, 14075, 14104
 000380 3734 3751      98          DC.W       14132, 14161, 14190, 14219, 14248
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

 00038A 37C4 37E1     99          DC.W        14276, 14305, 14334, 14363, 14392
 000394 3854 3871    100          DC.W        14420, 14449, 14478, 14507, 14536
 00039E 38E4 3901    101          DC.W        14564, 14593, 14622, 14651, 14680
 0003A8 3974 3991    102          DC.W        14708, 14737, 14766, 14795, 14824
 0003B2 3A04 3A21    103          DC.W        14852, 14881, 14910, 14939, 14968
 0003BC 3A94 3AB1    104          DC.W        14996, 15025, 15054, 15083, 15112
 0003C6 3B24 3B41    105          DC.W        15140, 15169, 15198, 15227, 15256
 0003D0 3BB4 3BD1    106          DC.W        15284, 15313, 15342, 15371, 15400
 0003DA 3C44 3C61    107          DC.W        15428, 15457, 15486, 15515, 15544
 0003E4 3CD4 3CF1    108          DC.W        15572, 15601, 15630, 15659, 15688
 0003EE 3D64 3D81    109          DC.W        15716, 15745, 15774, 15803, 15832
 0003F8 3DF4 3E11    110          DC.W        15860, 15889, 15918, 15947, 15976
 000402 0032 0106    111          DC.W               50,        262,        -262
                     112
 000408 CFFF CFFF    113          DC.W        -12289, -12289, -12289, -12289, -12289
 000412 CFFF CFFF    114          DC.W        -12289, -12289, -12289, -12289, -12289


 00041C CFFF CFFF    115          DC.W        -12289, -12289, -12289, -12289, -12289
 000426 CFFF CFFF    116          DC.W        -12289, -12289, -12289, -12289, -12289
 000430 CFFF CFFF    117          DC.W        -12289, -12289, -12289, -12289, -12289
 00043A CFFF CFFF    118          DC.W        -12289, -12289, -12289, -12289, -12289
 000444 CFFF CFFF    119          DC.W        -12289, -12289, -12289, -12289, -12289
 00044E CFFF CFFF    120          DC.W        -12289, -12289, -12289, -12289, -12289
 000458 CFFF CFFF    121          DC.W        -12289, -12289, -12289, -12289, -12289
 000462 CFFF CFFF    122          DC.W        -12289, -12289, -12289, -12289, -12289
 00046C CFFF CFFF    123          DC.W        -12289, -12289, -12289, -12289, -12289
 000476 CFFF CFFF    124          DC.W        -12289, -12289, -12289, -12289, -12289
 000480 CFFF CFFF    125          DC.W        -12289, -12289, -12289, -12289, -12289
 00048A CFFF CFFF    126          DC.W        -12289, -12289, -12289, -12289, -12289
 000494 CFFF CFFF    127          DC.W        -12289, -12289, -12289, -12289, -12289
 00049E CFFF CFFF    128          DC.W        -12289, -12289, -12289, -12289, -12289
 0004A8 CFFF CFFF    129          DC.W        -12289, -12289, -12289, -12289, -12289
 0004B2 CFFF CFFF    130          DC.W        -12289, -12289, -12289, -12289, -12289
 0004BC CFFF CFFF    131          DC.W        -12289, -12289, -12289, -12289, -12289
 0004C6 CFFF CFFF    132          DC.W        -12289, -12289, -12289, -12289, -12289
 0004D0 CFFF CFFF    133          DC.W        -12289, -12289, -12289, -12289, -12289
 0004DA CFFF CFFF    134          DC.W        -12289, -12289, -12289, -12289, -12289
 0004E4 CFFF CFFF    135          DC.W        -12289, -12289, -12289, -12289, -12289
 0004EE CFFF CFFF    136          DC.W        -12289, -12289, -12289, -12289, -12289
 0004F8 CFFF CFFF    137          DC.W        -12289, -12289, -12289, -12289, -12289
 000502 CFFF CFFF    138          DC.W        -12289, -12289, -12289, -12289, -12289
 00050C CFFF CFFF    139          DC.W        -12289, -12289, -12289, -12289, -12289
 000516 CFFF CFFF    140          DC.W        -12289, -12289, -12289, -12289, -12289
 000520 CFFF CFFF    141          DC.W        -12289, -12289, -12289, -12289, -12289
 00052A CFFF CFFF    142          DC.W        -12289, -12289, -12289, -12289, -12289
 000534 CFFF CFFF    143          DC.W        -12289, -12289, -12289, -12289, -12289
 00053E CFFF CFFF    144          DC.W        -12289, -12289, -12289, -12289, -12289
 000548 CFFF CFFF    145          DC.W        -12289, -12289, -12289, -12289, -12289
 000552 CFFF CFFF    146          DC.W        -12289, -12289, -12289, -12289, -12289
 00055C CFFF CFFF    147          DC.W        -12289, -12289, -12289, -12289, -12289
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

 000566 CFFF CFFF    148         DC.W     -12289, -12289, -12289, -12289, -12289
 000570 CFFF CFFF    149         DC.W     -12289, -12289, -12289, -12289, -12289
 00057A CFFF CFFF    150         DC.W     -12289, -12289, -12289, -12289, -12289
 000584 CFFF CFFF    151         DC.W     -12289, -12289, -12289, -12289, -12289
 00058E CFFF CFFF    152         DC.W     -12289, -12289, -12289, -12289, -12289
 000598 CFFF CFFF    153         DC.W     -12289, -12289, -12289, -12289, -12289
 0005A2 CFFF CFFF    154         DC.W     -12289, -12289, -12289, -12289, -12289
 0005AC CFFF CFFF    155         DC.W     -12289, -12289, -12289, -12289, -12289
 0005B6 CFFA CFF0    156         DC.W     -12294, -12304, -12319, -12340, -12367
 0005C0 CFBF CF65    157         DC.W     -12401, -12443, -12494, -12558, -12638
 0005CA CE36 CD7F    158         DC.W     -12746, -12929, -13357, -13498, -13594
 0005D4 CA9E CA63    159         DC.W     -13666, -13725, -13772, -13811, -13842
 0005DE C9D5 C9C2    160         DC.W     -13867, -13886, -13899, -13907, -13910
 0005E8 C9AC C9B3    161         DC.W     -13908, -13901, -13889, -13871, -13848
 0005F2 CA06 CA2B    162         DC.W     -13818, -13781, -13736, -13680, -13611
 0005FC CB2F CBAE    163         DC.W     -13521, -13394,      0,      0,      0
 000606 0000 0000    164         DC.W          0,      0,      0,      0, 13394
 000610 34D1 352B    165         DC.W     13521, 13611, 13680, 13736, 13781
 00061A 35FA 361B    166         DC.W     13818, 13848, 13871, 13889, 13901
 000624 3654 3656    167         DC.W     13908, 13910, 13907, 13899, 13886
 00062E 362B 3612    168         DC.W     13867, 13842, 13811, 13772, 13725
 000638 3562 351A    169         DC.W     13666, 13594, 13498, 13357, 12929
 000642 31CA 315E    170         DC.W     12746, 12638, 12558, 12494, 12443
 00064C 3071 304F    171         DC.W     12401, 12367, 12340, 12319, 12304


 000656 3006 3001    172         DC.W     12294, 12289, 12289, 12289, 12289
 000660 3001 3001    173         DC.W     12289, 12289, 12289, 12289, 12289
 00066A 3001 3001    174         DC.W     12289, 12289, 12289, 12289, 12289
 000674 3001 3001    175         DC.W     12289, 12289, 12289, 12289, 12289
 00067E 3001 3001    176         DC.W     12289, 12289, 12289, 12289, 12289
 000688 3001 3001    177         DC.W     12289, 12289, 12289, 12289, 12289
 000692 3001 3001    178         DC.W     12289, 12289, 12289, 12289, 12289
 00069C 3001 3001    179         DC.W     12289, 12289, 12289, 12289, 12289
 0006A6 3001 3001    180         DC.W     12289, 12289, 12289, 12289, 12289
 0006B0 3001 3001    181         DC.W     12289, 12289, 12289, 12289, 12289
 0006BA 3001 3001    182         DC.W     12289, 12289, 12289, 12289, 12289
 0006C4 3001 3001    183         DC.W     12289, 12289, 12289, 12289, 12289
 0006CE 3001 3001    184         DC.W     12289, 12289, 12289, 12289, 12289
 0006D8 3001 3001    185         DC.W     12289, 12289, 12289, 12289, 12289
 0006E2 3001 3001    186         DC.W     12289, 12289, 12289, 12289, 12289
 0006EC 3001 3001    187         DC.W     12289, 12289, 12289, 12289, 12289
 0006F6 3001 3001    188         DC.W     12289, 12289, 12289, 12289, 12289
 000700 3001 3001    189         DC.W     12289, 12289, 12289, 12289, 12289
 00070A 3001 3001    190         DC.W     12289, 12289, 12289, 12289, 12289
 000714 3001 3001    191         DC.W     12289, 12289, 12289, 12289, 12289
 00071E 3001 3001    192         DC.W     12289, 12289, 12289, 12289, 12289
 000728 3001 3001    193         DC.W     12289, 12289, 12289, 12289, 12289
 000732 3001 3001    194         DC.W     12289, 12289, 12289, 12289, 12289
 00073C 3001 3001    195         DC.W     12289, 12289, 12289, 12289, 12289
 000746 3001 3001    196         DC.W     12289, 12289, 12289, 12289, 12289
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

  000750 3001 3001     197          DC.W      12289, 12289, 12289, 12289, 12289
  00075A 3001 3001     198          DC.W      12289, 12289, 12289, 12289, 12289
  000764 3001 3001     199          DC.W      12289, 12289, 12289, 12289, 12289
  00076E 3001 3001     200          DC.W      12289, 12289, 12289, 12289, 12289
  000778 3001 3001     201          DC.W      12289, 12289, 12289, 12289, 12289
  000782 3001 3001     202          DC.W      12289, 12289, 12289, 12289, 12289
  00078C 3001 3001     203          DC.W      12289, 12289, 12289, 12289, 12289
  000796 3001 3001     204          DC.W      12289, 12289, 12289, 12289, 12289
  0007A0 3001 3001     205          DC.W      12289, 12289, 12289, 12289, 12289
  0007AA 3001 3001     206          DC.W      12289, 12289, 12289, 12289, 12289
  0007B4 3001 3001     207          DC.W      12289, 12289, 12289, 12289, 12289
  0007BE 3001 3001     208          DC.W      12289, 12289, 12289, 12289, 12289
  0007C8 3001 3001     209          DC.W      12289, 12289, 12289, 12289, 12289
  0007D2 3001 3001     210          DC.W      12289, 12289, 12289, 12289, 12289
  0007DC 3001 3001     211          DC.W      12289, 12289, 12289, 12289, 12289
  0007E6 3001 3001     212          DC.W      12289, 12289, 12289, 12289, 12289
  0007F0 3001 3001     213          DC.W      12289, 12289, 12289, 12289, 12289
  0007FA 3001 3001     214          DC.W      12289, 12289, 12289, 12289, 12289
  000804 000F 0333     215          DC.W             15,        819,       -819
                       216
  00080A CA09 CA09     217          DC.W      -13815, -13815, -13815, -13815, -13815
  000814 CA09 CA09     218          DC.W      -13815, -13815, -13815, -13815, -13815
  00081E CA09 CA09     219          DC.W      -13815, -13815, -13815, -13815, -13815
  000828 CA09 CA09     220          DC.W      -13815, -13815, -13815, -13815, -13815
  000832 CA09 CA09     221          DC.W      -13815, -13815, -13815, -13815, -13815
  00083C CA09 CA09     222          DC.W      -13815, -13815, -13815, -13815, -13815
  000846 CA09 CA09     223          DC.W      -13815, -13815, -13815, -13815, -13815
  000850 CA09 CA09     224          DC.W      -13815, -13815, -13815, -13815, -13815
  00085A CA09 CA09     225          DC.W      -13815, -13815, -13815, -13815, -13815
  000864 CA09 CA09     226          DC.W      -13815, -13815, -13815, -13815, -13815
  00086E CA09 CA09     227          DC.W      -13815, -13815, -13815, -13815, -13815
  000878 CA09 CA09     228          DC.W      -13815, -13815, -13815, -13815, -13815


  000882 CA09 CA09     229          DC.W      -13815, -13815, -13815, -13815, -13815
  00088C CA09 CA09     230          DC.W      -13815, -13815, -13815, -13815, -13815
  000896 CA09 CA09     231          DC.W      -13815, -13815, -13815, -13815, -13815
  0008A0 CA09 CA09     232          DC.W      -13815, -13815, -13815, -13815, -13815
  0008AA CA09 CA09     233          DC.W      -13815, -13815, -13815, -13815, -13815
  0008B4 CA09 CA09     234          DC.W      -13815, -13815, -13815, -13815, -13815
  0008BE CA09 CA09     235          DC.W      -13815, -13815, -13815, -13815, -13815
  0008C8 CA09 CA09     236          DC.W      -13815, -13815, -13815, -13815, -13815
  0008D2 CA09 CA09     237          DC.W      -13815, -13815, -13815, -13815, -13815
  0008DC CA09 CA09     238          DC.W      -13815, -13815, -13815, -13815, -13815
  0008E6 CA09 CA09     239          DC.W      -13815, -13815, -13815, -13815, -13815
  0008F0 CA09 CA09     240          DC.W      -13815, -13815, -13815, -13815, -13815
  0008FA CA09 CA09     241          DC.W      -13815, -13815, -13815, -13815, -13815
  000904 CA09 CA09     242          DC.W      -13815, -13815, -13815, -13815, -13815
  00090E CA09 CA09     243          DC.W      -13815, -13815, -13815, -13815, -13815
  000918 CA09 CA09     244          DC.W      -13815, -13815, -13815, -13815, -13815
  000922 CA09 CA09     245          DC.W      -13815, -13815, -13815, -13815, -13815
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

  00092C CA09 CA09    246          DC.W      -13815, -13815, -13815, -13815, -13815
  000936 CA09 CA09    247          DC.W      -13815, -13815, -13815, -13815, -13815
  000940 CA09 CA09    248          DC.W      -13815, -13815, -13815, -13815, -13815
  00094A CA09 CA09    249          DC.W      -13815, -13815, -13815, -13815, -13815
  000954 CA09 CA09    250          DC.W      -13815, -13815, -13815, -13815, -13815
  00095E CA09 CA09    251          DC.W      -13815, -13815, -13815, -13815, -13815
  000968 CA09 CA09    252          DC.W      -13815, -13815, -13815, -13815, -13815
  000972 CA09 CA09    253          DC.W      -13815, -13815, -13815, -13815, -13815
  00097C CA09 CA09    254          DC.W      -13815, -13815, -13815, -13815, -13815
  000986 CA09 CA09    255          DC.W      -13815, -13815, -13815, -13815, -13815
  000990 CA09 CA09    256          DC.W      -13815, -13815, -13815, -13815, -13815
  00099A CA09 CA09    257          DC.W      -13815, -13815, -13815, -13815, -13815
  0009A4 CA09 CA09    258          DC.W      -13815, -13815, -13815, -13815, -13815
  0009AE CA09 CA09    259          DC.W      -13815, -13815, -13815, -13815, -13815
  0009B8 CA09 CA09    260          DC.W      -13815, -13815, -13817, -13817, -13815
  0009C2 CA0D CA15    261          DC.W      -13811, -13803, -13794, -13781, -13766
  0009CC CA4B CA60    262          DC.W      -13749, -13728, -13704, -13678, -13648
  0009D6 CAD2 CAF7    263          DC.W      -13614, -13577, -13536, -13491, -13440
  0009E0 CBB8 CBF6    264          DC.W      -13384, -13322, -13252, -13174, -13084
  0009EA CD4D CDCC    265          DC.W      -12979, -12852, -12686, -12410,      0
  0009F4 0000 0000    266          DC.W           0,      0,      0,      0,      0
  0009FE 0000 0000    267          DC.W           0,      0,      0,      0,      0
  000A08 0000 0000    268          DC.W           0,      0,      0,      0,      0
  000A12 0000 0000    269          DC.W           0,      0,      0,      0,      0
  000A1C 0000 0000    270          DC.W           0,      0, 12410, 12686, 12852
  000A26 32B3 331C    271          DC.W      12979, 13084, 13174, 13252, 13322
  000A30 3448 3480    272          DC.W      13384, 13440, 13491, 13536, 13577
  000A3A 352E 3550    273          DC.W      13614, 13648, 13678, 13704, 13728
  000A44 35B5 35C6    274          DC.W      13749, 13766, 13781, 13794, 13803
  000A4E 35F3 35F7    275          DC.W      13811, 13815, 13817, 13817, 13815
  000A58 35F7 35F7    276          DC.W      13815, 13815, 13815, 13815, 13815
  000A62 35F7 35F7    277          DC.W      13815, 13815, 13815, 13815, 13815
  000A6C 35F7 35F7    278          DC.W      13815, 13815, 13815, 13815, 13815
  000A76 35F7 35F7    279          DC.W      13815, 13815, 13815, 13815, 13815
  000A80 35F7 35F7    280          DC.W      13815, 13815, 13815, 13815, 13815
  000A8A 35F7 35F7    281          DC.W      13815, 13815, 13815, 13815, 13815
  000A94 35F7 35F7    282          DC.W      13815, 13815, 13815, 13815, 13815
  000A9E 35F7 35F7    283          DC.W      13815, 13815, 13815, 13815, 13815
  000AA8 35F7 35F7    284          DC.W      13815, 13815, 13815, 13815, 13815
  000AB2 35F7 35F7    285          DC.W      13815, 13815, 13815, 13815, 13815


  000ABC 35F7 35F7    286          DC.W      13815, 13815, 13815, 13815, 13815
  000AC6 35F7 35F7    287          DC.W      13815, 13815, 13815, 13815, 13815
  000AD0 35F7 35F7    288          DC.W      13815, 13815, 13815, 13815, 13815
  000ADA 35F7 35F7    289          DC.W      13815, 13815, 13815, 13815, 13815
  000AE4 35F7 35F7    290          DC.W      13815, 13815, 13815, 13815, 13815
  000AEE 35F7 35F7    291          DC.W      13815, 13815, 13815, 13815, 13815
  000AF8 35F7 35F7    292          DC.W      13815, 13815, 13815, 13815, 13815
  000B02 35F7 35F7    293          DC.W      13815, 13815, 13815, 13815, 13815
  000B0C 35F7 35F7    294          DC.W      13815, 13815, 13815, 13815, 13815
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

  000B16 35F7 35F7    295          DC.W       13815, 13815, 13815, 13815, 13815
  000B20 35F7 35F7    296          DC.W       13815, 13815, 13815, 13815, 13815
  000B2A 35F7 35F7    297          DC.W       13815, 13815, 13815, 13815, 13815
  000B34 35F7 35F7    298          DC.W       13815, 13815, 13815, 13815, 13815
  000B3E 35F7 35F7    299          DC.W       13815, 13815, 13815, 13815, 13815
  000B48 35F7 35F7    300          DC.W       13815, 13815, 13815, 13815, 13815
  000B52 35F7 35F7    301          DC.W       13815, 13815, 13815, 13815, 13815
  000B5C 35F7 35F7    302          DC.W       13815, 13815, 13815, 13815, 13815
  000B66 35F7 35F7    303          DC.W       13815, 13815, 13815, 13815, 13815
  000B70 35F7 35F7    304          DC.W       13815, 13815, 13815, 13815, 13815
  000B7A 35F7 35F7    305          DC.W       13815, 13815, 13815, 13815, 13815
  000B84 35F7 35F7    306          DC.W       13815, 13815, 13815, 13815, 13815
  000B8E 35F7 35F7    307          DC.W       13815, 13815, 13815, 13815, 13815
  000B98 35F7 35F7    308          DC.W       13815, 13815, 13815, 13815, 13815
  000BA2 35F7 35F7    309          DC.W       13815, 13815, 13815, 13815, 13815
  000BAC 35F7 35F7    310          DC.W       13815, 13815, 13815, 13815, 13815
  000BB6 35F7 35F7    311          DC.W       13815, 13815, 13815, 13815, 13815
  000BC0 35F7 35F7    312          DC.W       13815, 13815, 13815, 13815, 13815
  000BCA 35F7 35F7    313          DC.W       13815, 13815, 13815, 13815, 13815
  000BD4 35F7 35F7    314          DC.W       13815, 13815, 13815, 13815, 13815
  000BDE 35F7 35F7    315          DC.W       13815, 13815, 13815, 13815, 13815
  000BE8 35F7 35F7    316          DC.W       13815, 13815, 13815, 13815, 13815
  000BF2 35F7 35F7    317          DC.W       13815, 13815, 13815, 13815, 13815
  000BFC 35F7 35F7    318          DC.W       13815, 13815, 13815, 13815, 13815

Errors=    0
```

```
FILE: SYSM1:pRBK         HEWLETT-PACKARD: 68000 Assembler                              Sat, 18 Jan 1986, 12:53     PAGE    1

LOCATION OBJECT CODE LINE     SOURCE LINE

                        1          "68000"
                        2          NAME "PUMULT MULTS. 2 16-BIT INTS, ADJUSTS TO PU FORM"
                        3
                        4           PROG
                        5           GLOBAL PUMULT                ; FUNCTION PUMULT
                        6           GLOBAL LPUMULT
                        7           GLOBAL APUDIV
  000000 4E56 FFFE      8 PUMULT    LINK      A6,#-2
  000004 302E 000A      9           MOVE.W    10[A6],D0
  000008 C1EE 0008     10           MULS      8[A6],D0          ; GET PRODUCT
  00000C E580          11           ASL.L     #2,D0             ; ALLOW FOR 1 LESS SIGN
  00000E 4840          12           SWAP      D0                 ; GET UPPER WORD
  000010 3D40 FFFE     13           MOVE.W    D0,-2[A6]
  000014 3E2E FFFE     14           MOVE.W    -2[A6],D7
  000018 4E5E          15           UNLK      A6
  00001A 4E75          16 RPUMULT   RTS
                       17
                       18
                       19          NAME " DIVIDES 2 16 BIT NO.S, ADJUSTS TO PU FORM"
  00001C 4E56 FFFE     20 APUDIV    LINK      A6,#-2
  000020 4280          21           CLR.L     D0
  000022 302E 000A     22           MOVE.W    10[A6],D0
  000026 4840          23           SWAP      D0
  000028 E480          24           ASR.L     #2,D0
  00002A 81EE 0008     25           DIVS      8[A6],D0
  00002E 3D40 FFFE     26           MOVE.W    D0,-2[A6]
  000032 3E2E FFFE     27           MOVE.W    -2[A6],D7
  000036 4E5E          28           UNLK      A6
  000038 4E75          29 RAPUDIV   RTS
                       30
                       31
                       32          NAME "MULTS. 2 16 BIT PU NUMBERS, PRODUCT = 32 BIT"
  00003A 4E56 FFFC     33 LPUMULT   LINK      A6,#-4
  00003E 302E 0008     34           MOVE.W    8[A6],D0
  000042 C1EE 000A     35           MULS      10[A6],D0
  000046 E580          36           ASL.L     #2,D0
  000048 2D40 FFFC     37           MOVE.L    D0,-4[A6]
  00004C 2E2E FFFC     38           MOVE.L    -4[A6],D7
  000050 4E5E          39           UNLK      A6
  000052 4E75          40 RLPUMULT  RTS

Errors=    0
```

```
FILE: SYSM2:pRBK        HEWLETT-PACKARD: 68000 Assembler                              Sat, 18 Jan 1986, 12:53     PAGE   1

LOCATION OBJECT CODE LINE     SOURCE LINE

                       1          "68000"
                       2          PROG
                       3          NAME "INTERRUPT AND VECTOR SET UP"
       <FFFFFFFF>      4 EIRQ1    EQU       $-1
                       5          GLOBAL    INTSET
                       6          EXTERNAL  PTMIRQ6,VECTOR6
                       7          EXTERNAL  IRQ3,VECTOR3
                       8          EXTERNAL  ACIAIRQ2,VECTOR2
  000000 4E56 FFFE     9 INTSET   LINK      A6,#-2
  000004 23FC         10          MOVE.L    #PTMIRQ6,VECTOR6    { LOAD AUTOVECTOR6 }
  00000E 23FC         11          MOVE.L    #IRQ3,VECTOR3       { LOAD AUTOVECTOR3 }
  000018 23FC         12          MOVE.L    #ACIAIRQ2,VECTOR2   { LOAD AUTOVECTOR2 }
  000022 46EE 000B    13          MOVE.W    8(A6),SR       { SET INTERRUPT LEVEL }
  000026 4E5E         14          UNLK      A6
  000028 4E75         15          RTS

Errors=    0
```

```
FILE: USER_LIB:pRBK     HEWLETT-PACKARD: 68000 Assembler                    Sat, 18 Jan 1986, 12:53     PAGE  1

LOCATION OBJECT CODE LINE     SOURCE LINE

                         1 "68000"
                         2        NAME "USER_LIB"
                         3 ;                      Zendprogram executes a 'return to
                         4 ;           monitor' trap. MONITOR_MESSAGE and JSR_ENTRY
                         5 ;           are not relevant but are defined to keep the
                         6 ;           linker happy.  Zstartprogram initializes
                         7 ;           pertinent address registers and sets up a stack.
                         8 ;
                         9        GLOBAL Zendprogram
                        10        GLOBAL Zstartprogram
                        11        GLOBAL MONITOR_MESSAGE
                        12        GLOBAL JSR_ENTRY
                        13        GLOBAL Zstack
 000000                 14 Zendprogram
 000000 4E4F            15          TRAP        #15
 000002 0000            16          DC.W        0000H
 000004                 17 MONITOR_MESSAGE
 000004                 18          DS.L        0
 000004                 19 JSR_ENTRY DS.L       0
                        20
                        21
 000004                 22 Zstartprogram
 000004 2A7C            23          MOVEA.L     #0,A5
 00000A 2C7C            24          MOVEA.L     #0,A6
 000010 4FF9            25          LEA         Zstack,A7
 000016 4ED0            26          JMP         (A0)
                        27          DATA
 000000                 28          DS.W         512
 000400                 29 Zstack   DS.W         1
                        30          END

Errors=    0
```

```
HP 64000 Linker                              Sat, 18 Jan 1986, 12:53          Page   1

FILE/PROG NAME              PROGRAM    DATA       COMMON     ABSOLUTE        DATE                TIME    COMMENTS
------------------------------------------------------------------------------------------------------------------------------
LOCO:pRBK                   0004000    0002000                               Sat, 18 Jan 1986, 12:45
IOLOCO:pRBK                 000453A    00021CA                               Sat, 18 Jan 1986, 12:47
DLOCO:pRBK                  0005432                                          Sat, 18 Jan 1986, 12:49
ELOCO:pRBK                  0005DF6    0002230                               Sat, 18 Jan 1986, 12:50
MLOCO:pRBK                  000661C    0002232                               Sat, 18 Jan 1986, 12:51
TLOCO:pRBK                  0006CBA                                          Sat, 18 Jan 1986, 12:52
ASMCURVE:pRBK                          000223C                               Sat, 18 Jan 1986, 12:52  DATA FOR SLIP CURVE(S)
USER_LIB:pRBK               0006E28    0002E42                               Sat, 18 Jan 1986, 12:53  USER_LIB
SYSM1:pRBK                  0006E40                                          Sat, 18 Jan 1986, 12:53  MULTS. 2 16 BIT PU NUM
SYSM2:pRBK                  0006E94                                          Sat, 18 Jan 1986, 12:53  INTERRUPT AND VECTOR S
next address                0006EBE    0003244


Libraries
A5_LIB:L68000
   A5Zmove:L68000           0006EBE                                          Wed,  7 Jul 1982, 11:33
next address                0006EE6    0003244


XFER address= 00043D0   Defined by LOCO
No. of passes through libraries=   1
absolute & link_com file name=LOCO:pRBK
Total# of bytes loaded= 000412A
```

Appendix A3

Display Computer Real-Time Software

DATDIS2	listing
MDAB, MDAB	Block Descriptions
MDAB MDABL	listings

DATA SET STORAGE BLOCK

CALL NAME: MDAB
CALL ADDRESS: 21C0

Program Memory

F***	
21C	
K***	K=7,8,9 Start Address
N_D	number of data channels
N_T	$N_{TMAX} = 12000/N_D$ no. data sets
CH1 addr	
CH2 addr	
CH3 addr	
.	
.	
.	
Chr addr	

running time: $(50n + 57)$ µs

Storage Memory

EXECUTION FLAG	00
	01
LOAD ADDR LOW B	0
HIGH B	03
N_D	04
N_D deer	05
N_T LOW B	06
HIGH B	07
	08

DATA SET "CONTINUOUS DOWNLOAD BLOCK

CALL NAME: MDABL
CALL ADDR: 21C3

Program Memory

F***	
21C3	
K***	K = 7,8,9 start addr
N_D	number of data channels

running time: 72 s

Storage Memory

EXECUTION FLAG	00
N_D	01
LOAD ADDR LOW B	02
HIGH B	03
	04

```
0000  1       "6809"
0000  1       $EXTENSIONS$
0000  1
0000  1  PROGRAM DATDIS;
0000  1
0000  1  (      *******DATA DISPLAY************RALPH KNOX*************
0000  1
0000  1         This program acquires data through PORT A of VIA1
0000  1         and sends it in cycles to PORT B.  The incoming data
0000  1         is sampled from the 68000  and the outgoing data is
0000  1         used to drive an oscilloscope.
0000  1
0000  1         REV1 : This display program has been enhanced to
0000  1         receive incoming data from the MICAS system via
0000  1         serial port 1 (ACIA1).  Data is then selected from
0000  1         both the MICAS system and the 68000.               )
0000  1
0000  1
0000  1  CONST    N = 8;         ( NO. OF CHANNELS IN DATA SET )
0000  1           NN = 4;        ( NOL OF CHANNELS IN MICAS DATA SET )
0000  1           M = 100;       ( NO. OF DATA SETS DISPLAYED )
0000  1           L = 800;       ( L = M * N = TOTAL NO. OF SAMPLES )
0000  1           T = 24883; ( SCOPE REFRESH (min~22 ms, use 27 ms for beats)
0000  1           T1FLAG = 6;    ( T1 INT. BIT IN IFR )
0000  1           CA1FLAG = 1;   ( CA1 INT. BIT IN IFR )
0000  1           RCVFLAG = 3;   ( receiver status in ACIA1 )
0000  1           START = 4000H; ( START ADDR. OF DATA SET ARRAY )
0000  1           UZERO = UNSIGNED_8(0);
0000  1           UONE = UNSIGNED_8(01);
0000  1
0000  1  TYPE     STATUSBITS = SET OF 0..7;
0000  1           EIGHT = UNSIGNED_8:
0000  1
0000  1  VAR      HOLDFLAG,ALLOCATE : BOOLEAN;
0002  1           I,C,BYTECNT,TCNT : INTEGER;
000A  1           DSCNT : INTEGER;
000C  1           STRTADDR : INTEGER;
000E  1           INPCHK : EIGHT;
000F  1           ONFLAG : ARRAY [1..N] OF BOOLEAN;
0017  1           MICASON : ARRAY [1..N] OF BOOLEAN;
001F  1           K : INTEGER;
0021  1           DUMREAD : EIGHT;
0022  1           VME,MICAS : EIGHT;
0024  1
0024  1          $GLOBVAR ON$
0024  1          $ORG 0CFE0H$           ( MEM MAPPED GLOBAL VARIABLES )
0024  1           ORB : EIGHT;
0024  1           IRA : EIGHT:          ( VIA PORTS )
0024  1           DDRB,DDRA : BYTE;
0024  1          $ORG 0CFE5H$
```

```
 51 0024  1            T1CH : INTEGER;  ( VIA AND TIMER CONTROL )
 52 0024  1            T1LH : INTEGER;  ( TIMER 1 )
 53 0024  1           $ORG 0CFEBH$
 54 0024  1            T2CL,T2CH : BYTE;  ( TIMER 2 )
 55 0024  1           $ORG 0CFEBH$
 56 0024  1            ACR,VPCR : BYTE;
 57 0024  1            IFR : STATUSBITS;
 58 0024  1            IER : BYTE;
 59 0024  1           $ORG 0CFEDH$

 60 0024  1            IFFR : BYTE;
 61 0024  1           $GLOBVAR OFF$
 62 0024  1
 63 0024  1           $ORG 0CFDBH$
 64 0024  1            AIAID1 : EIGHT;     ( ACIA1 PORT )
 65 0024  1            AIAST1 : STATUSBITS;
 66 0024  1            AIACMD1 : STATUSBITS;
 67 0024  1           $ORG 0CFDCH$
 68 0024  1            AIAID0 : CHAR;      ( ACIA0 PORT (NOT GLOBAL )
 69 0024  1            AIAST0 :STATUSBITS;
 70 0024  1            AIACMD0 : BYTE;
 71 0024  1           $ORG 0CFF1H$
 72 0024  1            ORA0 : STATUSBITS;
 73 0024  1           $ORG 0CFF3H$
 74 0024  1            DDRA0 : BYTE;
 75 0024  1           $ORG 4000H$
 76 0024  1            DATASET : ARRAY [1..2*L] OF EIGHT;
 77 0024  1           $END_ORG$
 78 0000  1
 79 0000  1
 80 0000  1  PROCEDURE INTSON; EXTERNAL;
 81 0000  1
 82 0000  1
 83 0000  1  $GLOBPROC ON$
 84 0000  1  $INTERRUPT ON$
 85 0000  1
 86 0000  1  PROCEDURE INTSERV;  ( senses keyboard generated
 87 0000  2                             control requests         )
 88 0000  2
 89 0000  2  VAR      CHRCTR : CHAR;
 90 0001  2           IX : INTEGER;
 91 0000  2
 92 0000  2  BEGIN
 93 0000  2     CHRCTR := AIAID0;
 94 0006  2
 95 0006  2               CASE CHRCTR OF
 96 0009  2
 97 0009  2          'R': HOLDFLAG := NOT HOLDFLAG;
 98 0014  2
 99 0014  2     '1','2','3','4',
100 0014  2     '5','6','7','8' : BEGIN
```

```
0014  2                          IX := ORD(CHRCTR) - 30H;
001E  2                          IF (IX <= NN) AND ALLOCATE THEN
002C  2                               MICASON [IX] := TRUE
003E  2                          ELSE
003E  2                               ONFLAG [IX] := NOT ONFLAG [IX];
0051  2
0051  2                          ALLOCATE := FALSE;
0054  2                        END;
0057  2
0057  2           'M' : ALLOCATE := TRUE;
005F  2
005F  2           'C' : BEGIN
005F  2                     ALLOCATE := FALSE;
0062  2                     FOR C:=1 TO N DO
0070  2                          BEGIN
0070  2                               ONFLAG [C] := FALSE;
007A  2                               MICASON [C] := FALSE;
0081  2                          END;

008F  2                END;
0092  2
0092  2           'I' : K := K + 1;
009E  2
009E  2           'D': IF K > 1 THEN K := K - 1;
00B0  2
00B0  2
00B0  2           OTHERWISE;     { IGNORE REQUEST }
00B0  2
00B0  2                END; { CASE }
00FF  2
00FF  2                REPEAT    { AN ASSERT STATEMENT (OPTIONAL) }
00FF  2                UNTIL NOT (7 IN AIASTO);
010B  2  END; { INTSERV }
0000  1
0000  1      $INTERRUPT OFF$
0000  1      $GLOBPROC OFF$
0000  1
0000  1
0000  1          { OUTPUTS ARRAY "DATASET" TO ORB QUICKLY }
0000  1  PROCEDURE WRDATSET(BADD: INTEGER );EXTERNAL;
0000  1
0000  1
0000  1
0000  1  PROCEDURE  VMEDAT(VAR DATA : EIGHT);
0113  2
0113  2      BEGIN
0113  2           VPCR := BYTE(0CCH); { SEND INT TO 68000}
0118  2
0118  2           REPEAT      { WAIT FOR READY FROM 68000 }
0118  2           UNTIL CAIFLAG IN IFR;
0122  2
```

```
0122  2          IFFR := 02H;  { RESET CA1 FLAG }
0127  2          DATA := IRA;
012E  2
012E  2          VPCR := BYTE(0CEH); { RESTORE INT LINE }
0133  2      END; { VMEDAT }
0000  1
0000  1
0000  1
0000  1  PROCEDURE MICASDAT(VAR DATA : EIGHT);
0139  2
0139  2      BEGIN
0139  2          ORA0 := ORA0 + STATUSBITS [0];
0141  2
0141  2          REPEAT   { wait for data input at ACIA1 }
0141  2          UNTIL RCVFLAG IN AIAST1;
014B  2
014B  2          ORA0 := ORA0 * STATUSBITS [1,2,3,4,5,6,7];
0153  2
0153  2          DATA := AIAIO1;
015A  2      END; { MICASDAT }
0161  1
0161  1
0161  1
0161  1  BEGIN
0161  1      DDRB := 0FFH;  { INITIALIZE TIMER AND PORTS }
0166  1      DDRA := 00H;
0169  1      DDRA0 := 0FFH;

016C  1      VPCR := 0CEH;  { CA2 HI,CB2 LOW ; CA1 -VE TRGGRD }
0171  1      ACR := 40H;
0176  1      ORA0 := ORA0 * STATUSBITS [1,2,3,4,5,6,7];  { set PA0 low }
017E  1      DUMREAD := AIAIO1;  { ensure receive flag cleared }
0184  1      T1LH := T;    { LOAD AND START TIMER1 }
018A  1      T1CH := T;
018D  1      AIACMD0 := 09H;   {   ENABLE ACIA0 INT'S. }
0192  1      FOR C := 1 TO N DO
01A0  1                 ONFLAG [C] := FALSE;
01B8  1      FOR C:= 1 TO NN DO
01C6  1                 MICASON [C] := FALSE;
01DE  1      INTSON;          { TURN INTS ON AND DEFINE VECTOR }
01E1  1      HOLDFLAG := FALSE;
01E4  1      ALLOCATE := FALSE;
01E7  1      DSCNT := 0;
01EC  1      TCNT := 1;
01F2  1      K := 1;
01F5  1      STRTADDR := START;   { DEFAULT VALUES FOR          }
01FB  1      DATASET [1] := UZERO;{ CASE WHEN K>1 ON STARTUP    }
01FE  1
01FE  1      I := 0;
0203  1      WHILE I = 0 DO   { LOOP INDEFINITELY }
020D  1      BEGIN
```

```
020D  1         IF NOT HOLDFLAG THEN  { CONTINUE IN REAL TIME.}
0216  1         BEGIN
0216  1              IF TCNT >= K THEN { GET NEW SAMPLES }
0220  1              BEGIN
0220  1                   TCNT := 1;
0226  1                   FOR BYTECNT := 1 TO N DO { LOAD AND STORE }
0231  1                   BEGIN
0231  1                        VMEDAT(VME);
0237  1                        IF BYTECNT <= NN THEN
0241  1                              MICASDAT(MICAS);
0247  1                        IF MICASON [BYTECNT] THEN
0255  1                             INPCHK := MICAS
025E  1                        ELSE
025E  1                             INPCHK := VME;
0264  1
0264  1                   { The first condition allows deletion of
0264  1                     unnecessary waveforms. The second condition
0264  1                     is needed since 0 is EOT detector.        }
0264  1
0264  1                        IF (NOT ONFLAG [BYTECNT])
0264  1                                  OR (INPCHK = UZERO) THEN
0282  1                            INPCHK := UONE;
0287  1
0287  1                        DATASET [DSCNT + BYTECNT] := INPCHK;
0297  1                        DATASET [DSCNT + BYTECNT + L] := INPCHK;
02AA  1                   END;
02BC  1                   DSCNT := DSCNT + N;
02C5  1                   IF DSCNT = L THEN  { RETURN TO START OF DS }
02CD  1                        DSCNT := 0;
02D3  1                   STRTADDR := START + DSCNT;  { DETERMINE }
02DC  1                   DATASET [DSCNT + L + 1] := UNSIGNED_8(0H);
02EC  1              END
02EF  1              ELSE   { SKIP SAMPLING AND JUST DISPLAY }
02EF  1                   TCNT := TCNT + 1;
02F8  1         END;
02F8  1

02F8  1         REPEAT
02F8  1              { DO NOTHING }
02F8  1         UNTIL T1FLAG IN IFR;  { UNTIL PERIOD T EXPIRES.}
0302  1         IFFR := 40H;    { RESET T1 FLAG }
0307  1
0307  1         WRDATSET(STRTADDR)  { DISPLAY INTERVAL }
030D  1    END;
0310  1 END.

End of compilation, number of errors=    0
```

```
FILE: MDAB:pRBK          HEWLETT-PACKARD: 8086 Assembler                              Tue, 18 Jun 1985, 15:05     PAGE    1

LOCATION OBJECT CODE LINE     SOURCE LINE

                       1          "8086"
                       2          NAME "MDAB CALL ADDR - 21C0 "
                       3
                       4 *        MICAS DATA SET STORAGE BLOCK - DISPLAY SYSTEM
                       5
                       6          PROG
                       7
  0000 8A05            8          MOV       AL,[DI]
  0002 0AC0            9          OR        AL,AL               ; TEST EXECUTION FLAG
  0004 7520           10          JNZ       NOTFIRST
  0006 C6050F         11          MOV  BYTE PTR[DI],#0FH        ; SET EXECUTION FLAG
  0009 8B4402         12          MOV       AX,[SI+2]           ; GET START ADDRESS
  000C 894502         13          MOV       [DI+2],AX           ; STORE START ADDRESS
  000F 8B4404         14          MOV       AX,[SI+4]           ; GET ND
  0012 884504         15          MOV       [DI+4],AL           ; STORE ND
  0015 884505         16          MOV       [DI+5],AL
  0018 0AC0           17          OR        AL,AL
  001A 743D           18          JZ        SKIP2               ; GO TO SKIP IF ND = 0
  001C 8B4406         19          MOV       AX,[SI+6]           ; GET NT
  001F 894506         20          MOV       [DI+6],AX           ; STORE NT
  0022 0BC0           21          OR        AX,AX
  0024 7433           22          JZ        SKIP2               ; GO TO SKIP IF NT = 0
  0026 BB0800         23 NOTFIRST MOV       BX,#08H             ; INIT CH ADDR PTR
  0029 53             24 LOOP1    PUSH      BX
  002A 8B18           25          MOV       BX,[BX][SI]         ; GET CH ADDR
  002C 8B07           26          MOV       AX,[BX]             ; GET CH DATA
  002E E83800         27          CALL      ADJUST
  0031 8B5D02         28          MOV       BX,[DI+2]           ; GET LOAD ADDR
  0034 8827           29          MOV       [BX],AH             ; STORE UPPER BYTE
  0036 43             30          INC       BX
  0037 895D02         31          MOV       [DI+2],BX
  003A FE4D05         32          DEC       BYTE PTR[DI+5]
  003D 7406           33          JZ        ENDLOOP             ; LOOPED THRU DATA SET?
  003F 5B             34          POP       BX
  0040 83C302         35          ADD       BX,#02H             ; INCR CH ADDR PTR
  0043 EBE4           36          JMP       LOOP1
  0045 8A4504         37 ENDLOOP  MOV       AL,[DI+4]           ; REINIT ND
  0048 884505         38          MOV       [DI+5],AL
  004B FF4D06         39          DEC       WORD PTR[DI+6]      ; DECR DATA SET CTR NT
  004E 7508           40          JNZ       SKIP1
  0050 32C0           41          XOR       AL,AL
  0052 8805           42          MOV       [DI],AL             ; IF NT = 0, RESET EX FLAG
  0054 B02E           43          MOV       AL,#2EH
  0056 8807           44          MOV       [BX],AL             ; PLACE '.' AT END OF LIST
  0058 5B             45 SKIP1    POP       BX
  0059 8A4504         46 SKIP2    MOV       AL,[DI+4]
  005C 98             47          CBW
  005D D1E0           48          SAL       AX,1                ; PTR TO NEXT BLOCK =
  005F 050A00         49          ADD       AX,#0AH             ; SI + 0AH + 2 * ND
  0062 03F0           50          ADD       SI,AX
  0064 83C708         51          ADD       DI,#08H
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   0067 FF24          52           JMP      WORD PTR[SI]
                      53
   0069 80FC0F        54 ADJUST    CMP      AH,#0FH
   006C 7614          55           JBE      VLDPOS              ; DETECT POS. NO. IN RANGE
   006E 80FCF0        56           CMP      AH,#0F0H
   0071 731B          57           JAE      VLDNEG              ; DETECT NEG. NO. IN RANGE


   0073 80FC80        58           CMP      AH,#80H
   0076 7205          59           JB       OVERPOS             ; DETECT POS. OVERFLOW
   0078 B400          60           MOV      AH,#00H             ; SET TO -VE LIMIT
   007A E91A00        61           JMP      FINI
   007D B4FF          62 OVERPOS   MOV      AH,#0FFH             ; SET TO +VE LIMIT
   007F E91500        63           JMP      FINI
   0082 80CC10        64 VLDPOS    OR       AH,#10H             ; ADJUST FOR DAC
   0085 D1E0          65           SHL      AX,1
   0087 D1E0          66           SHL      AX,1
   0089 D1E0          67           SHL      AX,1
   008B E90900        68           JMP      FINI
   008E 80E40F        69 VLDNEG    AND      AH,#0FH             ; ADJUST FOR DAC
   0091 D1E0          70           SHL      AX,1
   0093 D1E0          71           SHL      AX,1
   0095 D1E0          72           SHL      AX,1
   0097 C3            73 FINI      RET

Errors=    0
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                        1          "8086"
                        2          NAME "MDABL CALL ADDR - 21C3"
                        3
                        4 *        MICAS 'CONTINUOUS' DOWNLOAD BLOCK- DISPLAY SYSTEM
                        5
                        6          PROG
                        7
          <0000>        8 SERIO    EQU      0000H
          <0002>        9 SERSTA   EQU      0002H
                       10
  0000 8A05            11          MOV      AL,[DI]
  0002 0AC0            12          OR       AL,AL                        ; TEST EXECUTION FLAG
  0004 7513            13          JNZ      SKIPINIT
                       14
  0006 C6050F          15          MOV      BYTE PTR [DI],#0FH           ; SET EXECUTION FLAG
  0009 8B4402          16          MOV      AX,[SI+2]
  000C 894502          17          MOV      [DI+2],AX                    ; STORE LOAD ADDR
  000F 8B4404          18          MOV      AX,[SI+4]
  0012 884501          19          MOV      [DI+1],AL                    ; STORE ND
  0015 0AC0            20          OR       AL,AL                        ; TEST ND = 0
  0017 742D            21          JZ       SKIP20
                       22
  0019 51              23 SKIPINIT PUSH     CX
  001A B98400          24          MOV      CX,#0084H                    ; SERIAL PORT BASE ADDR
  001D 1E              25          PUSH     DS
  001E 8ED9            26          MOV      DS,CX                        ; CHANGE TO I/O SEG
  0020 C606020027      27          MOV      BYTE PTR DS:[SERSTA],#27H    ; INIT SERIAL PORT
  0025 A00200          28 PUTCHAR  MOV      AL,DS:[SERSTA]
  0028 2401            29          AND      AL,#01H
  002A 74F9            30          JZ       PUTCHAR                      ; ENSURE XMIT REG EMPTY
                       31
  002C 1F              32          POP      DS                           ; BACK TO PROG SEG
  002D 8B5D02          33          MOV      BX,[DI+2]
  0030 8A07            34          MOV      AL,[BX]                      ; GET LOAD ADDR
  0032 1E              35          PUSH     DS
  0033 8ED9            36          MOV      DS,CX                        ; BACK TO I/O SEG
  0035 A20000          37          MOV      DS:[SERIO],AL                ; SEND DATA
  0038 1F              38          POP      DS                           ; BACK TO PROG SEG
  0039 FE4502          39          INC      BYTE PTR [DI+2]
  003C FE4D01          40          DEC      BYTE PTR [DI+1]
  003F 7504            41          JNZ      SKIP10
                       42
  0041 32C0            43          XOR      AL,AL                        ; DATA SET XFER COMPLETE
  0043 8805            44          MOV      [DI],AL                      ; SO CLEAR INIT FLAG
  0045 59              45 SKIP10   POP      CX
  0046 83C608          46 SKIP20   ADD      SI,#08H
  0049 83C704          47          ADD      DI,#04H
  004C FF24            48          JMP      WORD PTR [SI]

Errors=    0
```

Appendix A4

Program Listings - Support Software

MLDR (see listings)

CURVEGEN

```
0000  1  "6809"
0000  1  $EXTENSIONS$
0000  1  PROGRAM MICASLOADER;
0000  1
0000  1  {  ***** MICAS LOADER ******************** RALPH KNOX *****
0000  1
0000  1     THIS IS A REVISED - ALL PASCAL VERSION - OF THE MICAS
0000  1     LOADER PROGRAM SENDMIC.
0000  1
0000  1     REV1: IT HAS BEEN  FURTHER REVISED TO HANDLE A VARIABLE
0000  1           LOAD ADDRESS.
0000  1
0000  1     REV2: NOW RECOGNIZES ADDRESS IDENTIFIERS SO MULTIPLE
0000  1           BLOCKS MAY BE LOADED.                          }
0000  1
0000  1  CONST    BLNK = BYTE(20H);
0000  1           DOT = BYTE(2EH);
0000  1           CR = BYTE(0DH);
0000  1           LF = BYTE(0AH);
0000  1           S = BYTE(53H);
0000  1           W = BYTE(57H);
0000  1           SEMCOL = BYTE(3BH);
0000  1
0000  1  TYPE     STATUSBITS = SET OF 0..7;
0000  1
0000  1  VAR      I,J,K : INTEGER;
0006  1           CMD : CHAR;
0007  1           CHR : BYTE;
0008  1         $ORG 4000H$ { FOR CONVENIENCE }
0008  1           ADDR : ARRAY [1..4] OF BYTE;
0008  1           LIST : ARRAY [0..10000] OF BYTE;
0008  1         $ORG 0CFD8H$     { ACIA1 ADDRESS }
0008  1           AIAIO1 : BYTE;
0008  1           AIAST1 : STATUSBITS;
0008  1         $ORG 0CFDCH$     { ACIA0 ADDRESS }
0008  1           AIAIO0 : CHAR;
0008  1           AIAST0 : STATUSBITS;
0008  1         $END_ORG$
0000  1
0000  1
0000  1  PROCEDURE TEACH;EXTERNAL;    { RETURNS CONTROL TO 6809 MONITOR }
0000  1
0000  1
0000  1  PROCEDURE GETCHAR0( VAR ICHAR0 : CHAR );    { GET A CHAR FROM
0000  2                                                ACIA0 (TER) PORT }
0000  2  CONST    RCVBIT = 3;
0003  2
0003  2  BEGIN
0003  2      REPEAT
0003  2      UNTIL RCVBIT IN AIAST0;
```

```
000D  2      ICHAR0 := AIAIO0
0014  2  END;
0000  1
0000  1
0000  1  PROCEDURE GETCHAR1( VAR ICHAR1 : BYTE);     { GET A CHAR FROM
0000  2                                                ACIA1 (AUX) PORT }
0000  2  CONST    RCVBIT = 3;
001A  2
001A  2  BEGIN
001A  2      REPEAT
001A  2      UNTIL RCVBIT IN AIAST1;
0024  2      ICHAR1 := AIAIO1
002B  2  END;
0000  1
0000  1
0000  1  PROCEDURE PUTCHAR1( OCHAR1: BYTE);     { OUTPUT A CHAR TO
0000  2                                           ACIA1 (AUX) PORT  }
0000  2  CONST    XMTRBIT = 4;
0031  2
0031  2  BEGIN
0031  2      REPEAT
0031  2      UNTIL XMTRBIT IN AIAST1;
003B  2      AIAIO1 := OCHAR1
0041  2  END;
0000  1
0000  1
0000  1  PROCEDURE WAIT;                     {  GET AND DISCARD CHAR'S AT
0000  2                                         ACIA1 PORT UNTIL EOT      }
0000  2  CONST    RCVBIT = 3;
0000  2           EOT = BYTE(04H);
0000  2
0000  2  VAR      DISCARD : BYTE;
0043  2
0043  2  BEGIN
0043  2      REPEAT
0043  2           REPEAT
0043  2           UNTIL RCVBIT IN AIAST1;
004D  2           DISCARD := AIAIO1
0053  2      UNTIL DISCARD = EOT
0057  2  END;
005D  1
005D  1
005D  1
005D  1  BEGIN    { MICASLOADER }
005D  1      GETCHAR0(CMD); { INPUT COMMAND FROM KEYBOARD }
0062  1      CASE CMD OF
0068  1
0068  1      'I': BEGIN     { LOAD HP FILE TO 6809 }
0068  1                FOR J := 1 TO 4 DO    { RECORD START ADDRESS }
0076  1                     BEGIN
```

```
101 0076  1                         REPEAT  ( DISCARD EXTRANEOUS INFO )
102 0076  1                              GETCHAR1(CHR)
103 007B  1                         UNTIL (CHR<>BLNK)AND(CHR<>CR)AND(CHR<>LF);
104 00A1  1                         ADDR [J] := CHR
105 00B1  1                     END;
106 00C2  1               I := 0;
107 00C7  1               REPEAT
108 00C7  1                    REPEAT    ( DISCARD EXTRANEOUS INFO )
109 00C7  1                          GETCHAR1(CHR)
110 00CD  1                    UNTIL (CHR<>BLNK)AND(CHR<>CR)AND(CHR<>LF);
111 00F3  1                    LIST [I] := CHR;    ( STORE PRGM AS ARRAY )
112 0103  1                    I := I + 1
113 010C  1               UNTIL CHR = DOT     ( A "." INDICATES END OF FILE )
114 0113  1          END; ( I )
115 0116  1
116 0116  1    '0' : BEGIN     ( LOAD PRGM IN 6809 MEM TO MICAS )
117 0116  1               PUTCHAR1(S);
118 011B  1               PUTCHAR1(W);

119 0120  1               PUTCHAR1(BLNK);     ( INTERACT WITH MICAS MONITOR )
120 0125  1               FOR J := 1 TO 4 DO  ( SPECIFY LOAD ADDRESS )
121 0133  1                    PUTCHAR1(ADDR [J]);
122 0151  1               PUTCHAR1(CR);
123 0156  1               WAIT;
124 0159  1               I := 0;
125 015E  1               WHILE LIST [I] <> DOT DO
126 016E  1                    BEGIN
127 016E  1                         FOR K := 1 TO 4 DO  (SEND CODE TO MICAS)
128 017C  1                              BEGIN
129 017C  1                                   PUTCHAR1(LIST [I]);
130 0189  1                                   I := I + 1
131 0192  1                              END;
132 01A3  1                         PUTCHAR1(SEMCOL);
133 01A8  1                         WAIT
134 01AB  1                    END;
135 01AD  1               PUTCHAR1(CR)
136 01B2  1          END  ( O )
137 01B5  1     END;  ( CASE )
138 01C4  1     TEACH
139 01C7  1  END.     ( MICASLOADER )

End of compilation, number of errors=    0
```

```
    1  1   1:D      1 "HOST"
    2  1   1:D      1 PROGRAM GRAPH(INPUT,OUTPUT,ASMFILE);
    3  1   1:D      1 $EXTENSIONS$
    4  1   1:D      1
    5  1   1:D      1 CONST    M = 34;
    6  1   1:D      1          N = 74;
    7  1   1:D      1          MAXPU = 16384;
    8  1   1:D      1          LENGTH = 255;
    9  1   1:D      1
   10  1   1:D      1 TYPE     GRID = ARRAY [-M..M,-N..N] OF CHAR;
   11  1   1:D      1          PUINT = INTEGER;
   12  1   1:D      1          LIST = ARRAY [- LENGTH..(LENGTH-1)] OF INTEGER;
   13  1   1:D      1
   14  1   1:D      1          PROFILE = RECORD
   15  1   1:D      1                       SLOPE : INTEGER;
   16  1   1:D      1                       LIN : INTEGER;
   17  1   1:D      1                       NLIN : INTEGER;
   18  1   1:D      1                       TABLE : LIST;
   19  1   1:D      1                    END;
   20  1   1:D      1
   21  1   1:D      1 VAR      H,R1,R2,MTA,MTB,XMAX : REAL;
   22  1   1:D     13          NH,NR1,NR2,NMTA,NMTB,NXMAX : REAL;
   23  1   1:D     25          M1 : INTEGER;
   24  1   1:D     27          PFTYPE : CHAR;
   25  1   1:D     28          OPENFILE,LAST : CHAR;
   26  1   1:D     30          NEGQUAD : CHAR;
   27  1   1:D     31          XLIN,XINC,THLIN,MPA,MPB,THA,THB : REAL;
   28  1   1:D     45          XTA,XTB,YTB : REAL;
   29  1   1:D     51          X1,Y1,X2,Y2 : REAL;
   30  1   1:D     59          U,V : REAL;
   31  1   1:D     63          V1,V2 : REAL;
   32  1   1:D     67          MAX : PUINT;
   33  1   1:D     69          NMAX : PUINT;
   34  1   1:D     71          I,J : INTEGER;
   35  1   1:D     75          K,L : INTEGER;
   36  1   1:D     79          GRAPH : GRID;
   37  1   1:D  10360          CURVESET : ARRAY [1..3] OF PROFILE;
   38  1   1:D  13438          INDEX :INTEGER;
   39  1   1:D  13440          SHOW : CHAR;
   40  1   1:D  13441
   41  1   1:D  13441          ASMFILE : TEXT;
   42  1   1:D  13572
   43  1   1:D  13572
   44  1   1:D  13572
   45  1   2:D      1 PROCEDURE GETDAT;
   46  1   2:D      1
   47  1   2:C      0 BEGIN
   48  1   2:C      0
   49  1   2:C      0     WRITELN;
   50  1   2:C     15     WRITELN('Is the specified file to be appended (Y)');
```

```
 51  1   2:C    79      WRITELN('or created new (N).');
 52  1   2:C   122      READ(OPENFILE);
 53  1   2:C   140      WRITELN;
 54  1   2:C   155      WRITELN('Is this the last profile');
 55  1   2:C   203      WRITELN('for this file? (Y/N)');
 56  1   2:C   247      READLN;
 57  1   2:C   257      READ(LAST);

 58  1   2:C   275      WRITELN;
 59  1   2:C   290      IF (OPENFILE = 'Y') AND (LAST = 'Y') THEN
 60  1   2:C   301      WRITELN('                                                 LAST APPEND');
 61  1   2:C   382      WRITELN('Enter profile number :');
 62  1   2:C   428      READ(INDEX);
 63  1   2:C   443      WRITELN('                                                 PROFILE NUMBER :  ',
 64  1   2:C   526                  INDEX:2);
 65  1   2:C   542      WRITELN('Enter profile type - A :  GM profile');
 66  1   2:C   602      WRITELN('                   - B :  BBC profile');
 67  1   2:C   663      READLN;
 68  1   2:C   673      READ(PFTYPE);
 69  1   2:C   691      WRITELN('                                                  PROFILE TYPE :  ',
 70  1   2:C   774                  PFTYPE:1);
 71  1   2:C   787      WRITELN;
 72  1   2:C   802      WRITELN('Display graph? (Y/N)');
 73  1   2:C   846      READLN;
 74  1   2:C   856      READ(SHOW);
 75  1   2:C   875      WRITELN;
 76  1   2:C   890      WRITELN('Enter linear slope M1 :');
 77  1   2:C   937      READ(M1);
 78  1   2:C   951      WRITELN('                                                            M1 :  ',
 79  1   2:C  1034                  M1:3);
 80  1   2:C  1049      WRITELN;
 81  1   2:C  1064      WRITELN('*****POSITIVE QUADRANT PARAMATERS*****');
 82  1   2:C  1126      WRITELN;
 83  1   2:C  1141      WRITELN('Enter height H of positive linear region :');
 84  1   2:C  1207      READ(H);
 85  1   2:C  1219      WRITELN('                                                             H :  ',
 86  1   2:C  1302                  H:2:2);
 87  1   2:C  1318      WRITELN('Enter PU radius R1 of circle 1 :');
 88  1   2:C  1374      READ(R1);
 89  1   2:C  1386      WRITELN('                                                            R1 :  ',
 90  1   2:C  1469                  R1:2:2);
 91  1   2:C  1485 IF PFTYPE = 'A' THEN
 92  1   2:C  1491 BEGIN
 93  1   2:C  1491      WRITELN('Enter PU radius R2 of circle 2 :');
 94  1   2:C  1547      READ(R2);
 95  1   2:C  1559      WRITELN ('                                                           R2 :  ',
 96  1   2:C  1641                  R2:2:2);
 97  1   2:C  1657      WRITELN('Enter slope MTA at tangent point :');
 98  1   2:C  1715      READ(MTA);
 99  1   2:C  1727      WRITELN('                                                           MTA :  ',
100  1   2:C  1810                  MTA:2:2);
```

```
101  1    2:C  1826 END
102  1    2:C  1826 ELSE
103  1    2:C  1828 BEGIN
104  1    2:C  1828     WRITELN('Enter slope MTB of 2nd linear region :');
105  1    2:C  1890     READ(MTB);
106  1    2:C  1902     WRITELN('                                                     MTB :  ',
107  1    2:C  1985               MTB:2:2);
108  1    2:C  2001 END;
109  1    2:C  2001
110  1    2:C  2001     WRITELN('Enter maximum PU slip XMAX :');
111  1    2:C  2053     READ(XMAX);
112  1    2:C  2065     WRITELN('                                                    XMAX :  ',
113  1    2:C  2148              XMAX:2:2);
114  1    2:C  2164     WRITELN;

115  1    2:C  2179     WRITELN('Negative quadrant plot required? (Y/N)');
116  1    2:C  2241     READLN;
117  1    2:C  2251     READ(NEGQUAD);
118  1    2:C  2269
119  1    2:C  2269 IF NEGQUAD = 'Y' THEN
120  1    2:C  2275 BEGIN
121  1    2:C  2275     WRITELN;
122  1    2:C  2290     WRITELN('*****NEGATIVE QUADRANT PARAMETERS*****');
123  1    2:C  2352     WRITELN;
124  1    2:C  2367
125  1    2:C  2367     WRITELN('Enter height H of negative linear region :');
126  1    2:C  2433     READ(NH);
127  1    2:C  2445     WRITELN('                                                       H :  ',
128  1    2:C  2528               H:2:2);
129  1    2:C  2544     WRITELN('Enter PU radius R1 of circle 1 :');
130  1    2:C  2600     READ(NR1);
131  1    2:C  2612     WRITELN('                                                      R1 :  ',
132  1    2:C  2695               R1:2:2);
133  1    2:C  2711 IF PFTYPE = 'A' THEN
134  1    2:C  2717 BEGIN
135  1    2:C  2717     WRITELN('Enter PU radius R2 of circle 2 :');
136  1    2:C  2773     READ(NR2);
137  1    2:C  2785     WRITELN ('                                                     R2 :  ',
138  1    2:C  2867               R2:2:2);
139  1    2:C  2883     WRITELN('Enter slope MTA at tangent point :');
140  1    2:C  2941     READ(NMTA);
141  1    2:C  2953     WRITELN('                                                     MTA :  ',
142  1    2:C  3036               MTA:2:2);
143  1    2:C  3052 END
144  1    2:C  3052 ELSE
145  1    2:C  3054 BEGIN
146  1    2:C  3054     WRITELN('Enter slope MTB of 2nd linear region :');
147  1    2:C  3116     READ(NMTB);
148  1    2:C  3128     WRITELN('                                                     MTB :  ',
149  1    2:C  3211               MTB:2:2);
150  1    2:C  3227 END;
```

```
151  1   2:C  3227
152  1   2:C  3227      WRITELN('Enter maximum PU slip IMAX :');
153  1   2:C  3279      READ(NXMAX);
154  1   2:C  3291      WRITELN('                                                                    IMAX :  ',
155  1   2:C  3374                IMAX:2:2);
156  1   2:C  3390
157  1   2:C  3390 END;
158  1   2:C  3390
159  1   2:C  3390 END; (GETDAT )
160  1   2:C  3412
161  1   2:C  3412
162  1   3:D     1 PROCEDURE INIT(VAR GRPH: GRID);
163  1   3:D     2
164  1   3:D     2 VAR      I,J : INTEGER;
165  1   3:D     6
166  1   3:C     0 BEGIN
167  1   3:C     0      FOR I := -M TO M DO
168  1   3:C    29           FOR J := -N TO N DO
169  1   3:C    58                GRPH [ I,J ] := ' ';
170  1   3:C   148 END;
171  1   3:C   164

172  1   3:C   164
173  1   4:D     1 PROCEDURE STARTFILE;
174  1   4:D     1
175  1   4:C     0 BEGIN
176  1   4:C     0
177  1   4:C     0      WRITELN(ASMFILE,'          "68000"');
178  1   4:C    41      WRITELN(ASMFILE,'          NAME "DATA FOR SLIP CURVE(S)."');
179  1   4:C   105      WRITELN(ASMFILE,'          DATA');
180  1   4:C   143      WRITELN(ASMFILE,'          GLOBAL   CURVESET');
181  1   4:C   194      WRITELN(ASMFILE);
182  1   4:C   209      WRITELN(ASMFILE,'CURVESET');
183  1   4:C   242
184  1   4:C   242 END;  { STARTFILE }
185  1   4:C   254
186  1   4:C   254
187  1   4:C   254
188  1   5:D     1 PROCEDURE CIRCLE(XVAR,XFIX,YFIX,RADIUS: REAL; VAR RT1,RT2: REAL);
189  1   5:D    11
190  1   5:D    11
191  1   5:D    11 VAR  ATERM,BTERM : REAL;
192  1   5:D    15      CTERM,S : REAL;
193  1   5:D    19
194  1   5:D    19
195  1   5:D    19
196  1   6:D     3      FUNCTION YCONST(XVAR,XFIX,YFIX,RADIUS : REAL) : REAL;
197  1   6:D    11
198  1   6:C     0      BEGIN
199  1   6:C     0          YCONST := (XVAR * XVAR) - (2 * XFIX * XVAR) + (XFIX * XFIX)
200  1   6:C    40                           + (YFIX * YFIX) - (RADIUS * RADIUS);
```

```
 201  1   6:C    69      END;
 202  1   6:C    82
 203  1   6:C    82
 204  1   6:C    82
 205  1   7:D     3      FUNCTION ROOT(A,B,C : REAL) : REAL;
 206  1   7:D     9
 207  1   7:C     0      BEGIN
 208  1   7:C     0          ROOT := SQRT( B * B - 4.0 * A * C);
 209  1   7:C    37      END;
 210  1   7:C    50
 211  1   7:C    50
 212  1   7:C    50
 213  1   5:C     0 BEGIN
 214  1   5:C     0
 215  1   5:C     0     ATERM := 1.0;
 216  1   5:C    10     BTERM := -2.0 * YFIX;
 217  1   5:C    29     CTERM := YCONST(XVAR,XFIX,YFIX,RADIUS);
 218  1   5:C    53     S := ROOT(ATERM,BTERM,CTERM);
 219  1   5:C    73     RT1 := (- BTERM + S) / (2.0 * ATERM);
 220  1   5:C   103     RT2 := (- BTERM - S) / (2.0 * ATERM);
 221  1   5:C   133
 222  1   5:C   133 END; { CIRCLE }
 223  1   5:C   146
 224  1   5:C   146
 225  1   5:C   146
 226  1   8:D     1 PROCEDURE DISPLAY(U,V : REAL);
 227  1   8:D     5
 228  1   8:D     5 CONST    SCALEX = 34.0;

 229  1   8:D     5          SCALEY = 74.0;
 230  1   8:D     5
 231  1   8:D     5 VAR      A,B : INTEGER;
 232  1   8:D     9
 233  1   8:C     0 BEGIN
 234  1   8:C     0     A := ROUND(SCALEX * U);
 235  1   8:C    18     B := ROUND(SCALEY * V);
 236  1   8:C    36     GRAPH [A,B] := '*';
 237  1   8:C    83
 238  1   8:C    83 END; { DISPLAY }
 239  1   8:C    96
 240  1   8:C    96
 241  1   8:C    96
 242  1   1:C     0 BEGIN { MAIN PRGM }
 243  1   1:C     0
 244  1   1:C     0
 245  1   1:C     0     GETDAT;
 246  1   1:C    91
 247  1   1:C    91     IF OPENFILE = 'Y' THEN
 248  1   1:C    97          APPEND(ASMFILE);
 249  1   1:C   107
 250  1   1:C   107     IF OPENFILE = 'N' THEN
```

```
1  1:C  113          BEGIN
1  1:C  113               REWRITE(ASMFILE);
1  1:C  123               STARTFILE;
1  1:C  125          END;
1  1:C  125
1  1:C  125     INIT(GRAPH);
1  1:C  129
1  1:C  129  WITH CURVESET [INDEX] DO
1  1:C  151  BEGIN
1  1:C  151
1  1:C  151     { Positive Quadrant algorithm }
1  1:C  151
1  1:C  151
1  1:C  151     XLIN := H / M1;
1  1:C  167     XINC := XMAX / (LENGTH + 1);
1  1:C  185     THLIN := ARCTAN(XLIN / H);
1  1:C  202     X1 := XLIN + R1 * COS(THLIN);
1  1:C  226     Y1 := H - R1 * SIN(THLIN);
1  1:C  250     U := 0;
1  1:C  257     K := 0;
1  1:C  263
1  1:C  263     SLOPE := M1;
1  1:C  272     LIN := ROUND(XLIN * MAXPU);
1  1:C  292
1  1:C  292     REPEAT
1  1:C  292          V := M1 * U;
1  1:C  310          DISPLAY(U,V);
1  1:C  320          TABLE [K] := 0;
1  1:C  349          U := U + XINC;
1  1:C  364          K := K + 1;
1  1:C  375     UNTIL (U > XLIN) OR (U > XMAX);
1  1:C  398
1  1:C  398     K := K - 1;  { this moves intermediate values }
1  1:C  409                  { to the last point instead of the next }
1  1:C  409

1  1:C  409          CASE PFTYPE OF
1  1:C  413
1  1:C  413    'A' : BEGIN
1  1:C  413               MPA := -1 / MTA;
1  1:C  430               THA := ARCTAN(MPA);
1  1:C  442               XTA := X1 + R1 * COS(THA);
1  1:C  466               X2 := X1 + (R1+R2) * COS(THA);
1  1:C  496               Y2 := Y1 + (R1+R2) * SIN(THA);
1  1:C  526
1  1:C  526               IF U <= XMAX THEN
1  1:C  538                    REPEAT
1  1:C  538                         CIRCLE(U,X1,Y1,R1,V1,V2);
1  1:C  560                         V := V1;
1  1:C  568                         DISPLAY(U,V);
1  1:C  578                         TABLE [K] := ROUND(V * MAXPU);
```

```
301  1   1:C   618                        U := U + XINC;
302  1   1:C   633                        K := K + 1;
303  1   1:C   644                   UNTIL (U > XTA) OR (U > XMAX);
304  1   1:C   667
305  1   1:C   667              IF U <= XMAX THEN
306  1   1:C   679                   REPEAT
307  1   1:C   679                        CIRCLE(U,X2,Y2,R2,V1,V2);
308  1   1:C   701                        V := V2;
309  1   1:C   709                        DISPLAY(U,V);
310  1   1:C   719                        TABLE [K] := ROUND(V * MAXPU);
311  1   1:C   759                        U := U + XINC;
312  1   1:C   774                        K := K + 1;
313  1   1:C   785                   UNTIL (U > X2) OR (U > XMAX);
314  1   1:C   808
315  1   1:C   808              IF U <= XMAX THEN
316  1   1:C   820                   REPEAT
317  1   1:C   820                        DISPLAY(U,V);
318  1   1:C   830                        TABLE [K] := ROUND(V * MAXPU);
319  1   1:C   870                        U := U + XINC;
320  1   1:C   885                        K := K + 1;
321  1   1:C   896                   UNTIL U >= XMAX;
322  1   1:C   908
323  1   1:C   908        END;
324  1   1:C   910
325  1   1:C   910   'B' : BEGIN
326  1   1:C   910              IF MTB = 0 THEN
327  1   1:C   921                   BEGIN
328  1   1:C   921                        XTB := X1;
329  1   1:C   929                        YTB := Y1 + R1;
330  1   1:C   944                   END
331  1   1:C   944              ELSE
332  1   1:C   946                   BEGIN
333  1   1:C   946                        MPB := -1 / MTB;
334  1   1:C   963                        THB := ARCTAN(MPB);
335  1   1:C   975                        IF THB < 0 THEN
336  1   1:C   986                             BEGIN
337  1   1:C   986                                  XTB := X1 - R1 * COS(THB);
338  1   1:C  1010                                  YTB := Y1 - R1 * SIN(THB);
339  1   1:C  1034                             END
340  1   1:C  1034                        ELSE
341  1   1:C  1036                             BEGIN
342  1   1:C  1036                                  XTB := X1 + R1 * COS(THB);
343  1   1:C  1060                                  YTB := Y1 + R1 * SIN(THB);
344  1   1:C  1084                             END;
345  1   1:C  1084                   END;
346  1   1:C  1084
347  1   1:C  1084              IF U <= XMAX THEN
348  1   1:C  1096                   REPEAT
349  1   1:C  1096                        CIRCLE(U,X1,Y1,R1,V1,V2);
350  1   1:C  1118                        V := V1;
```

```
 1   1:C  1126                          DISPLAY(U,V);
 1   1:C  1136                          TABLE [K] := ROUND(V * MAXPU);
 1   1:C  1176                          U := U + XINC;
 1   1:C  1191                          K := K + 1;
 1   1:C  1202                     UNTIL (U > XTB) OR (U > XMAX);
 1   1:C  1225
 1   1:C  1225                IF U <= XMAX THEN
 1   1:C  1237                     REPEAT
 1   1:C  1237                          V := MTB * U - MTB * XTB + YTB;
 1   1:C  1270                          DISPLAY(U,V);
 1   1:C  1280                          TABLE [K] := ROUND(V * MAXPU);
 1   1:C  1320                          U := U + XINC;
 1   1:C  1335                          K := K + 1;
 1   1:C  1346                     UNTIL U >= XMAX;
 1   1:C  1358         END;
 1   1:C  1360
 1   1:C  1360     END; { CASE }
 1   1:C  1376
 1   1:C  1376     { Negative plot algorithm }
 1   1:C  1376 WRITELN('NEGATIVE SECTION');
 1   1:C  1419
 1   1:C  1419 IF NEGQUAD = 'Y' THEN
 1   1:C  1425 BEGIN
 1   1:C  1425     XMAX := NXMAX;
 1   1:C  1433     H := NH;
 1   1:C  1441     R1 := NR1;
 1   1:C  1449     MTB := NMTB;
 1   1:C  1457     R2 := NR2;
 1   1:C  1465     MTA := NMTA;
 1   1:C  1473
 1   1:C  1473
 1   1:C  1473     XLIN := H / M1;
 1   1:C  1489     XINC := XMAX / (LENGTH + 1);
 1   1:C  1507     THLIN := ARCTAN(XLIN / H);
 1   1:C  1524     X1 := XLIN + R1 * COS(THLIN);
 1   1:C  1548     Y1 := H - R1 * SIN(THLIN);
 1   1:C  1572     U := 0;
 1   1:C  1579     K := 0;
 1   1:C  1585
 1   1:C  1585
 1   1:C  1585     NLIN := - ROUND(XLIN * MAXPU);
 1   1:C  1607
 1   1:C  1607     REPEAT
 1   1:C  1607          V := M1 * U;
 1   1:C  1625          DISPLAY(-U,-V);
 1   1:C  1641          TABLE [-K] := 0;
 1   1:C  1672          U := U + XINC;
 1   1:C  1687          K := K + 1;
 1   1:C  1698     UNTIL (U > XLIN) OR (U > XMAX);

 1   1:C  1721
```

```
1   1:C  1721   K := K - 1;
1   1:C  1732
1   1:C  1732
1   1:C  1732        CASE PFTYPE OF
1   1:C  1736
1   1:C  1736  'A' : BEGIN
1   1:C  1736             MPA := -1 / MTA;
1   1:C  1753             THA := ARCTAN(MPA);
1   1:C  1765             XTA := X1 + R1 * COS(THA);
1   1:C  1789             X2 := X1 + (R1+R2) * COS(THA);
1   1:C  1819             Y2 := Y1 + (R1+R2) * SIN(THA);
1   1:C  1849
1   1:C  1849             IF U <= XMAX THEN
1   1:C  1861                  REPEAT
1   1:C  1861                       CIRCLE(U,X1,Y1,R1,V1,V2);
1   1:C  1883                       V := V1;
1   1:C  1891                       DISPLAY(-U,-V);
1   1:C  1907                       TABLE [-K] := - ROUND(V * MAXPU);
1   1:C  1951                       U := U + XINC;
1   1:C  1966                       K := K + 1;
1   1:C  1977                  UNTIL (U > XTA) OR (U > XMAX);
1   1:C  2000
1   1:C  2000             IF U <= XMAX THEN
1   1:C  2012                  REPEAT
1   1:C  2012                       CIRCLE(U,X2,Y2,R2,V1,V2);
1   1:C  2034                       V := V2;
1   1:C  2042                       DISPLAY(-U,-V);
1   1:C  2058                       TABLE [-K] := - ROUND(V * MAXPU);
1   1:C  2102                       U := U + XINC;
1   1:C  2117                       K := K + 1;
1   1:C  2128                  UNTIL (U > X2) OR (U > XMAX);
1   1:C  2151
1   1:C  2151             IF U <= XMAX THEN
1   1:C  2163                  REPEAT
1   1:C  2163                       DISPLAY(-U,-V);
1   1:C  2179                       TABLE [-K] := - ROUND(V * MAXPU);
1   1:C  2223                       U := U + XINC;
1   1:C  2238                       K := K + 1;
1   1:C  2249                  UNTIL U > XMAX;
1   1:C  2261
1   1:C  2261       END;
1   1:C  2263
1   1:C  2263  'B' : BEGIN
1   1:C  2263             IF MTB = 0 THEN
1   1:C  2274                  BEGIN
1   1:C  2274                       XTB := X1;
1   1:C  2282                       YTB := Y1 + R1;
1   1:C  2297                  END
1   1:C  2297             ELSE
1   1:C  2299                  BEGIN
1   1:C  2299                       MPB := -1 / MTB;
```

```
452  1   1:C  2316                         THB := ARCTAN(MPB);
453  1   1:C  2328                         IF THB < 0 THEN
454  1   1:C  2339                              BEGIN
455  1   1:C  2339                                   XTB := X1 - R1 * COS(THB);
456  1   1:C  2363                                   YTB := Y1 - R1 * SIN(THB);

457  1   1:C  2387                              END
458  1   1:C  2387                         ELSE
459  1   1:C  2389                              BEGIN
460  1   1:C  2389                                   XTB := X1 + R1 * COS(THB);
461  1   1:C  2413                                   YTB := Y1 + R1 * SIN(THB);
462  1   1:C  2437                              END;
463  1   1:C  2437                    END;
464  1   1:C  2437
465  1   1:C  2437               IF U <= XMAX THEN
466  1   1:C  2449                    REPEAT
467  1   1:C  2449                         CIRCLE(U,X1,Y1,R1,V1,V2);
468  1   1:C  2471                         V := V1;
469  1   1:C  2479                         DISPLAY(-U,-V);
470  1   1:C  2495                         TABLE [-K] := - ROUND(V * MAXPU);
471  1   1:C  2539                         U := U + XINC;
472  1   1:C  2554                         K := K + 1;
473  1   1:C  2565                    UNTIL (U > XTB) OR (U > XMAX);
474  1   1:C  2588
475  1   1:C  2588               IF U <= XMAX THEN
476  1   1:C  2600                    REPEAT
477  1   1:C  2600                         V := MTB * U - MTB * XTB + YTB;
478  1   1:C  2633                         DISPLAY(-U,-V);
479  1   1:C  2649                         TABLE [-K] := - ROUND(V * MAXPU);
480  1   1:C  2693                         U := U + XINC;
481  1   1:C  2708                         K := K + 1;
482  1   1:C  2719                    UNTIL U > XMAX;
483  1   1:C  2731
484  1   1:C  2731          END;
485  1   1:C  2733
486  1   1:C  2733     END; { CASE }
487  1   1:C  2748 END; { NEGQUAD }
488  1   1:C  2748
489  1   1:C  2748 IF SHOW = 'Y' THEN
490  1   1:C  2755     FOR J := N DOWNTO -N DO
491  1   1:C  2789          BEGIN
492  1   1:C  2789               FOR I := -M TO M DO
493  1   1:C  2824                    WRITE(GRAPH [I,J]);
494  1   1:C  2904               WRITELN;
495  1   1:C  2922          END;
496  1   1:C  2948
497  1   1:C  2948 WRITELN(ASMFILE,'        DC.W      ',SLOPE,',',LIN,',',NLIN);
498  1   1:C  3058
499  1   1:C  3058 FOR L := - LENGTH TO (LENGTH - 1) DO
500  1   1:C  3102     BEGIN
501  1   1:C  3102          IF L MOD 5 = 0 THEN
```

```
 502  1   1:C  3114              BEGIN
 503  1   1:C  3114                  WRITELN(ASMFILE);
 504  1   1:C  3132                  WRITE(ASMFILE,'         DC.W        ');
 505  1   1:C  3175              END;
 506  1   1:C  3175          IF (L MOD 5 = 4) OR (L = LENGTH - 1) THEN
 507  1   1:C  3200              WRITE(ASMFILE,TABLE [L]:5)
 508  1   1:C  3246          ELSE
 509  1   1:C  3248              WRITE(ASMFILE,TABLE [L]:5,', ');
 510  1   1:C  3308      END;
 511  1   1:C  3334
 512  1   1:C  3334  END; ( WITH )
 513  1   1:C  3334

 514  1   1:C  3334 END. ( GRAPH )

End of compilation, number of errors=    0
```

Appendix A5

MICAS Microslip Program

- Listing

```
FD00
2007
AE0A                                            NMOT
AE1C                                            NLOCO
E900

F004
2022
E900
E902

FD08
2031
E902
5FF0            WA1             2.5%            5MIN
E904

FD0C
20A0
E904
E906

FD10
20C1
E908
E90A
E90C
E90E

FD14
20A6
E90C
5FF2            WA2             0.0%
E910

FD18
2007
AE02                                            TMOT
E912
E914
FD1C
2031
E914
5FF4            WA3             0.15%           TDMIN
E916

FD20
2091
E918
E906
E918
```

```
FD24
2052
E918
5FF2          WA2          0.0%
5FFA          WA6          2.8%          SLOPMIN
E91A

FD28
2082
E904
E910
E91C

FD2C
2094
0003
E906
E91C
E91E
0000
E920

FD30
2085
E91C
E916
E922

FD34
2085
E904
E910
E924

FD38
2052
E924
E926
AE02                                     TMOT
E926

FD3C
203A
AE1E                                     SLPOT
E928
5FF2          WA2          0.0%
E92A
E92C
E92E

FD40
2082
E92E
E924
E930
```

```
FD44
2085
E924
E92E
E90A

FD48
2013
5FF6          WA4          0.4%          SLSTEP
5FF8
E932
FD4C
2007
E928
E932
E934

FD50
2091
E922
E930
E936

FD54
2052
E936
E91A
E934
E938

FD58
2052
E924
E938
E93A
E93C

FD5C
2004
E93C
5FF6          WA4          0.4%          SLSTEP
E93A

FD60
203A
E93C
5FFA          WA6          2.8%          SLOPMIN
5FF2          WA2          0.0%
E93E
E940
E942
```

```
c       *****   DCOS5   *****   MAY 1985   *****   R. KNOX

c               DCOS5 is the "DEC Communications Operating
c               System Ver 5"  which complements the MCOS5
c               operating system residing at the Micas system.
c               DCOS5 implements flexible, interactive remote
c               control of the Micas end, allowing user-
c               controlled data acquisition. Incoming data is
c               continuously appended to a data file which is
c               periodically written to the disc.  The data
c               file and a parameter file are created to be
c               used by the graphing program "GRAPH".

c               Program execution takes the form of menu
c               requests. The menu options reflect the various
c               modes that the Micas program MCOS3 can be in.
c               There are two main modes of operation:

c                       1)  Continuous Data Acquisition where
c                       data is constantly downloaded and
c                       appended to a  file.  This process may
c                       be initiated or halted at will.

c                       2)  Fault Data Acquisition where after
c                       a reset command, data is stored
c                       continuously in memory residing within
c                       the Micas system. A fault condition at
c                       the Micas halts further storage and a
c                       download command at this end stores
c                       data to a file.

c               Other menu requests allow displaying of either
c               data file, setting up of the Micas system, and
c               exiting from the program.

c               Setting up the Micas system is accomplished by
c               inputting information the user specifies and
c               packaging it into several commands which are
c               then sent to the Micas system.  The following
c               commands may be sent:

c               N:  specifies the number of active "continuous"
c                   channels.
c               P:  specifies the number of active "fault"
c                   channels.
c               A:  assigns a data address to a continuous
c                   channel.
c               E:  assigns a data address to a fault channel.
c               K:  defines the relative sample rate of the
c                   continuous system.  (Sample rate = K * Ts.)
c               D:  specifies the delay which is to follow a
c                   fault condition.
c               C:  establishes Micas execution mode by sending
c                   one of four command characters.
```

```
c               MCOS parameters NOR(=)1, IC(=)C, MicB(=)b,
c                               NT(=)NT
        parameter (NOR = 4)
        parameter (IC = 31,MicB = 250)
        parameter (NT = 39)

        parameter (IIR = 82,IIS = 83,IID = 68,IIC = 67)
        parameter (IIN = 78,IIP = 80,IIK = 75,IIE = 69)
        parameter (III = 73,IIG = 71)

        Integer KeyBd,chr
        Integer In,Ip,IK,NumRec,RecLen
        Integer BKS,IStr,NF
        Integer IntRec(500),SSR,IRec
        Integer IX,FRecL
        Character*1 Ctrl,Ackn
        Character*2 Cn,Cp,Cdly
        Character*4 Addr(10),Faddr(10),ChrK
        Character*255 RecTyp(50)
        Logical fault

c               Define protocol characters.
        Character*1 STX,ETX,ENQ,ACK
        Byte dSTX,dETX,dENQ,dACK
        Equivalence (dSTX,STX),(dETX,ETX),(dENQ,ENQ),(dACK,ACK)

        dSTX = 02
        dETX = 03
        dENQ = 05
        dACK = 06

        IStr = 0
        fault = .false.
        NF = NT + 1

c               Serial Port tt10 is assigned logical device 3
        call assign(3,'tt10:')

        type 01
01      format('1',T20,'Micas Communications Operating System
     + Ver 5.0')


c               Ensure no data transmission from Micas.
        call CmdC(STX,'S',ETX)

c               Check for acknowledgement.
        read(3,02) Ackn
02      format(A1)
        if (Ackn .NE. ACK) then
                type 03
03              format('0','****ERROR: Cmd C:S not
     + accepted by Micas.****')
        end if
```

```
c               Initialization parameters for Micas
c               and this program are accepted.

        call UserIO(In,Cn,Addr,IK,ChrK,Ip,Cp,Faddr,Cdly)


c               Commands are sent according to UserIO

        call SndCmd(In,Cn,Addr,IK,ChrK,Ip,Cp,Faddr,Cdly,
     +               STX,ETX,ACK)


c               Basic parameters are stored in a file
c               to be accessed later by graphing and data
c               processing routines. Calculation are also made.

        call Parms(NOR,IC,NF,In,IK,Ip,SSR,IRecL,IBKS,MicB,
     +              FRecL)


c               Display command menu.
        call UsrCmd


c               Begin active execution structure.

c               Check for a keyboard input.
120     KeyBd = igtchr(chr)


c            exit if input = 'E'
        if (KeyBd .NE. IIE) then

c                 Skip if no input.
             if (KeyBd .NE. -1) then

                  if (KeyBd .EQ. III) then

c                            Repeat set up.
                       call UserIO(In,Cn,Addr,IK,ChrK,Ip,Cp,
     +                              Faddr,Cdly)

                       call SndCmd(In,Cn,Addr,IK,ChrK,Ip,Cp,
     +                              Faddr,Cdly,STX,ETX,ACK)

                       call Parms(NOR,IC,NF,In,IK,Ip,SSR,IRecL,
     +                             IBKS,MicB,FRecL)

                  else
                       if (KeyBd .EQ. IIR) then

c                                 Do continuous acquisition.
                          call BlockR(IBKS,IRecL,RecTyp,NOR,
     +                                IStr,fault,IIS,STX,ETX,
     +                                ACK,ENQ,SSR,IntRec)
```

```
                       else
                            if (KeyBd .EQ. IID) then

c                                    Has fault occurred?
                                 if (fault) then

c                                          Do fault download
c                                          block.
                                      call BlockD(RecTyp,NF,
     +                                            STX,ETX,ACK,
     +                                            IntRec,FRecL)
                                 end if
                            else
                                 if(KeyBd .EQ. IIN) then

c                                          Display continuous
c                                          data.
                                      call BlockN(IBKS,IntRec,
     +                                            IRecL,IStr,
     +                                            IIK)
                                 else
                                      if (KeyBd .EQ. IIP) then

c                                               Display fault
c                                               data.
                                        call BlockP(NF,
     +                                                IntRec,
     +                                                FRecL,IIK)
                                      else
                                           if (KeyBd .EQ. IIC)
     +                                       then

c                                                   Resume
c                                                   data
c                                                   monitoring.
                                             call BlockC(fault,
     +                                                        STX,
     +                                                        ETX,
     +                                                        ACK)
                                           end if
                                      end if
                                 end if
                            end if
                       end if
                  end if

                  call UsrCmd
             end if

c                 Loop back and poll for command request.
             go to 120

        end if

        end
        subroutine UserIO(In,Cn,Addr,IK,ChrK,Ip,Cp,Faddr,Cdly)

        integer In,IK,Ip
        character*2 Cn,Cp,Cdly
        character*4 Addr(10),ChrK,Faddr(10)

c               This subroutine handles the interactive user
c               inputs and outputs.  The input information is
c               used to send control commands to the Micas
c               system and also to adjust control parameters
c               in the active part of this program.

        integer ChNo,Idly

        type 10
10      format('1',T20,'MICAS SETUP')

        type 11
11      format(////T5,'Continuous Acquisition System')

        type 12
12      format(///'$',T5,'Specify number of data channels:  ')
        accept 14,In
14      format(I2)
        encode(2,16,Cn) In
16      format(Z2.2)

        type 18
18      format(//T5,'Specify data address of each channel.'/
     +     T5,'(Micas addresses are in HEX with capital
     + letters.)'//)

        do 500 ChNo = 1,In
                type 20,ChNo
20              format('$',T20,'Channel ',I2,':   ')
                accept 22,Addr(ChNo)
22              format(A4)
500     continue

        type 204,Addr
204     format(T10,'   ',10A5)

        type 24
24      format(//T5,'Specify sample rate multiplier K.'/'$',
     +          T5,'(Ts = 20 ms for K = 1) :  ')
        accept 26,IK
26      format(I5)
        encode(5,28,ChrK) IK
28      format(Z4.4)


        type 30
30      format(//T5,'Fault Detection Acquistion System')
```

```
      type 32
32    format(///'$',T5,'Specify number of data channels:  ')
      accept 34,Ip
34    format(I2)
      encode(2,36,Cp)Ip
36    format(Z2.2)

      type 38
38    format(//T5,'Specify data HEX addresses of each channel
     +.')

      do 510 ChNo = 1,Ip
      type 40,ChNo
40    format('$',T20,'Channel',I2,':  ')
      accept 42,Faddr(ChNo)
42    format(A4)
510   continue

      type 44
44    format(//T5,'Specify number of cycles delay required
     + after'//'$',T5,'fault trap before data storage is
     + discontinued:  ')
      accept 46,Idly
46    format(I3)
      encode(2,48,Cdly)Idly
48    format(Z2.2)

      return
      end


      subroutine SndCmd(In,Cn,Addr,IK,ChrK,Ip,Cp,Faddr,Cdly,
     +                  STX,ETX,ACK)

      Integer In,IK,Ip
      Character*1 STX,ETX,ACK
      Character*2 Cn,Cp,Cdly
      Character*4 Addr(10),ChrK,Faddr(10)


c     This routine sends the commands that set up Micas
c     according to the specifications obtained from 'UserIO'.

      Integer Num
      Character*2 CNum


c               Set up Micas for continuous data
c               transmission by specifying number of
c               channels, data addresses, and sample rate.


      if (In .NE. 0) then
           call CmdN(STX,Cn,ETX,ACK)
```

```
             do 100 Num = 1,In
                  encode(2,1000,CNum) Num
1000              format(Z2.2)
                  type 1001,Addr(Num)
1001              format(' ',A4)
                  call CmdA(STX,CNum,Addr(Num),ETX,ACK)
100          continue
        end if

        if (IK .NE. 0) then
             call CmdK(STX,ChrK,ETX,ACK)
        end if


c               Set up Micas for fault monitoring
c               by specifying number of channels,
c               data addresses, and delay parameter.

        if (Ip .NE. 0) then
             Call CmdP(STX,Cp,ETX,ACK)

             do 110 Num = 1, Ip
                 encode(2,1010,CNum)Num
1010             format(Z2.2)
                 call CmdE(STX,CNum,Faddr(Num),ETX,ACK)
110          continue
        end if

        call CmdD(STX,Cdly,ETX,ACK)

        return
        end



        subroutine Parms(NOR,IC,NF,In,IK,Ip,SSR,IRecL,IBKS,
     +                  MicB,FRecL)

        Integer NF,In,IK,Ip,SSR,IRecL,IBKS,FRecL

c       System constants are stored in a parameter file for
c       later use by graphing and data processing routines.
c       Record size and iteration count values are calculated.

        Integer IX

        open(unit = 4, file = 'dcos5.prm', status = 'new',
     +       dispose = 'save', access = 'sequential',
     +       organization = 'sequential', blocksize = 1,
     +       form = 'unformatted', recl = 24,
     +       recordtype = 'variable', initialsize = 1,
     +       extendsize = 1)

        write(4) NOR,IC,NF,In,IK,Ip
        close(unit = 4, dispose = 'save')

        SSR = IC * In
        IRecL = NOR * SSR
        IBKS = IRecL / 512  + 1

        IX = MicB / (2 * Ip)
        FRecL = IX * Ip

        return
        end



        subroutine UsrCmd

 c      This routine executes 'isetup' which allows
 c      asynchronous keyboard inputs using 'igtchr'.
 c      Also a command menu is displayed showing
 c      possible asynchronous command options.

        Integer Idum,Jdum

        Jdum = isetup(Idum)

        type 50
 50     format('1',T20,'MCOS3 COMMAND MENU')

        type 52
 52     format(//T5,'Choose one of the following command
+ options'/T5,'by typing the appropriate letter.')

        type 54
 54     format(//T5,'R',T10,'Start continuous system running.'
      + //T5,'S',T10,'Stop continuous system.'//T5,'D',T10,
      + 'Download fault trap data.'//T5,'C',T10,'Resume fault
      + detection monitoring.'//T5,'N',T10,'Display continuous
      + data storage file.'//T5,'P',T10,'Display fault data
      + storage file.'//T5,'K',T10,'Kill data display.'//T5,
      + 'E',T10,'Exit DCOS3.'//T5,'I',T10,'Set up MICAS.')

        return
        end



        subroutine BlockD(InpRec,NF,STX,ETX,ACK,FIntgr,FRecL)

        Character*1 STX,ETX,ACK
        Character*255 InpRec(NF)
        Integer NF,FRecL,FIntgr(FRecL)

 c              Operational block initiated by command 'D'
 c              which requests a download of fault data.
 c              This data is stored in the file 'dcos5.ud2'.
```

```
      Integer K,p,a
      Character*255 AckRec

c           Open a sequential file.

      open(unit = 4, file = 'dcos5.ud2', status = 'new',
    +     dispose = 'save', access = 'sequential',
    +     organization = 'sequential', blocksize = 1,
    +     form = 'unformatted', recl = FRecL,
    +     recordtype = 'variable', initialsize = 1,
    +     extendsize = 1)

c           Initiate data transmission.
      call CmdC(STX,'D',ETX)

      type 3050
3050  format('0',T20,'Downloading fault data . . .')

c           input data
      do 300 K=1,NF
            read(3,3000) InpRec(K)
3000        format(A255)
300   continue

      do 310 n=2,NF
            p = 1
            a = 1
320         if (InpRec(n)(p:p) .NE. ' ') then
                  decode(2,3010,InpRec(n)(p:(p+1)))
    +                   FIntgr(a)
3010              format(Z2.2)
                  if (FIntgr(a) .GT. 127) then
                        FIntgr(a) = FIntgr(a) - 256
                  end if
                  a = a + 1
                  p = p + 2
                  go to 320
            end if
            write (4) FIntgr
310   continue

c           close file (store data on disc)
      close(unit = 4, dispose = 'save')


      return
      end
```

```
        subroutine BlockC(fault,STX,ETX,ACK)

        Logical fault
        Character*1 STX,ETX,ACK

c               Operational block 'C' resumes continuous
c               fault data acquisition by sending the
c               appropriate command to the Micas.

        Character*255 AckRec

c               resume data monitoring
        call CmdC(STX,'C',ETX)

c               detect acknowledge from Micas
        read(3,5500) AckRec
5500    format(A255)

        if (AckRec(:1) .NE. ACK) then
                type 5510
5510            format('0','***ERROR--reset attempt failed.***'
     +)
        else
                fault = .false.
                type 5520
5520            format('0',T20,'Fault Monitoring Resumed.')
        end if

        return
        end


        subroutine BlockR(IBKS,IRecL,InpRec,NOR,KCnt,fault,IIS,
     +                    STX,ETX,ACK,ENQ,SSR,Intgr)

        Integer IBKS,IRecL,KCnt,SSR
        Integer Intgr(IRecL)
        Character*1 STX,ETX,ACK,ENQ
        Character*255 InpRec(NOR)
        Logical fault

c               Operational block 'R' requests the continuous
c               downloading of data from the Micas.  Data is
c               stored in blocks consisting of BKS input
c               records. Input record length depends upon the
c               number of channels being downloaded.  Data is
c               stored in 'dcos5.udl'.

        Logical AckFlg,Msg
        Integer J
        Integer aptr,p,m,I
        open(unit = 4, file = 'dcos5.udl', status = 'new',
     +       dispose = 'save', access = 'sequential',
     +       organization = 'sequential', blocksize = IBKS,
     +       form = 'unformatted', recl = IRecL, recordtype =
     +       'variable', initialsize = 1, extendsize = 1)

        AckFlg = .false.
        Msg = .false.


c               Begin continuous transmission.
        call CmdC(STX,'R',ETX)

        type 2500
2500    format('0',T20,'Begin continuous data acquisition . . .
     +'/)

c               Store data.

        KCnt = 1
200     J = 1

210     if (J .LE. NOR) then

                read(3,2000) InpRec(J)
2000            format(A255)

                type 2010,InpRec(J)(:1)
2010            format(' ',A1)

                if (InpRec(J)(:1) .EQ. ACK) then
                        AckFlg = .true.
                else
                        if (InpRec(J)(:1) .EQ. ENQ) then
                                fault = .true.
                                Msg = .true.
                        else
                                J = J + 1
                        end if
                end if

                if (AckFlg .AND. J .EQ. 1) then
                        go to 280
                else
                        go to 210
                end if

        end if

        aptr = 1
        do 250 n = 1, NOR
                p = 1
                if (InpRec(n)(p:p) .EQ. 'S') then
                        p = p + 1
```

```
260                     if (InpRec(n)(p:p) .NE. ' ') then
                          decode(2,2040,InpRec(n)(p:(p+1)))
     +                           Intgr(aptr)
2040                      format(Z2.2)
                          if (Intgr(aptr) .GT. 127) then
                                Intgr(aptr) = Intgr(aptr) - 256
                          end if
                          p = p + 2
                          aptr = aptr + 1
                          go to 260
                        end if
                else
                        do 270 m = aptr, (aptr + SSR - 1)
                                Intgr(m) = 500
270                     continue
                        aptr = aptr + SSR
                        type 2050
2050                    format('0','***ERROR - Corrupted Data
     + Record.')
                end if
250     continue

c       type 2060,Intgr
c2060   format(12(2X,I3))

        write(4) Intgr


c               close and reopen file to secure data on disc.
        close(unit = 4, dispose = 'save')

        open(unit = 4, file = 'dcos5.udl', status = 'old',
     +  dispose = 'save', access = 'append',
     +  organization = 'sequential', blocksize = IBKS,
     +  form = 'unformatted', recl = IRecL,
     +  recordtype = 'variable', initialsize = 1,
     +  extendsize = 1)


c               When applicable, indicate that a fault
c               has occurred.
        if (Msg) then
                type 2030
2030            format('0',T5,'MICAS fault has occurred.'/)
                Msg = .false.
        end if

c               Poll for stop transmission command.
        KeyBd = igtchr(chr)

        if (KeyBd .NE. -1) then
                if (KeyBd .EQ. IIS) then
                        call CmdC(STX,'S',ETX)
                end if
        end if
```

```
c               Halt data input and storage when
c               ACK is received.

280     if (.not. AckFlg) then
                KCnt = KCnt + 1
                go to 200
        else
                AckFlg = .false.
                close(unit = 4, dispose = 'save')
        end if


        return
        end




        subroutine BlockN(IBKS,GetRec,IRecL,Cnt,IIK)

        Integer GetRec(IRecL),IBKS,IRecL,Cnt

c               Operational Block 'N' displays sequentially
c               the "continuous" data that has been stored
c               in 'dcos5.ud1'.  File displaying may be halted
c               anytime by the control command 'K'.

        Integer n,chr

        open(unit = 4, file = 'dcos5.ud1', status = 'old',
     +       dispose = 'save', access = 'sequential',
     +       organization = 'sequential', blocksize = IBKS,
     +       form = 'unformatted', recl = IRecL,
     +       recordtype = 'variable', initialsize = 1,
     +       extendsize = 1)

        do 400 n = 1,Cnt

                read(4) GetRec

                type 4003
4003            format('0')

                type 4005,GetRec
4005            format(12(1X,I4))



c                       Poll for 'kill' command.

                KeyBd = igtchr(chr)

                if (KeyBd .NE.-1) then
                        if (KeyBd .EQ. IIK) then
                                go to 420
                        end if
                end if
```

```
400     continue

420     close(unit = 4, dispose = 'save')

        return
        end


        subroutine BlockP(Cnt,GetRec,FRecL,IIK)

        Integer Cnt,FRecL,GetRec(FRecL)

c               Operational block 'P' displays file 'dcos5.ud2'
c               containing fault downloaded data.  File
c               displaying may be aborted by executing the 'K'
c               kill command.

        Integer p


        open(unit = 4, file = 'dcos5.ud2', status = 'old',
     +       dispose = 'save', access = 'sequential',
     +       organization = 'sequential', blocksize = 1,
     +       form = 'unformatted', recl = FRecL,
     +       recordtype = 'variable', initialsize = 1,
     +       extendsize = 1)

        do 500 p = 1,Cnt-1

                read(4) GetRec

                type 5004
5004            format('0')

                type 5005,GetRec
5005            format(12(1X,I4))


c                               Poll for 'kill' command.

                KeyBd = igtchr(chr)

                if (KeyBd .NE. -1) then
                        if (KeyBd .EQ. IIK) then
                                go to 510
                        end if
                end if

500     continue

510     close(unit = 4, dispose = 'save')

        return
        end
        subroutine CmdN(B,Cn,E,A)

        Character*1 B,E,A
        Character*2 Cn

c               'N' command subroutine indicates
c               to Micas the number of active
c               "continuous" data channels .

        Character*1 Ackn

        Write(3,6000) B,Cn,E
6000    format('+',A1,'N ',A2,A1)

        read(3,6002) Ackn
6002    format(A1)

        if (Ackn .NE. A) then
             type 6004
6004         format('0','****ERROR: Cmd N not accepted by
     + Micas.****')
        end if

        return
        end




        subroutine CmdA(B,CNum,HexAdd,E,A)

        Character*1 B,E,A
        Character*2 CNum
        Character*4 HexAdd

c               'A' command subroutine defines a
c               Hex 'continuous' data address for
c               a given channel number.  This
c               information is sent to the Micas.

        Character*1 Ackn

        Write(3,6010) B,CNum,HexAdd,E
6010    format('+',A1,'A ',A2,' ',A4,A1)

        read(3,6012) Ackn
6012    format(A1)

        if (Ackn .NE. A) then
             type 6014
6014         format('0','****ERROR: Cmd A not accepted by
     + Micas.****')
        end if

        return
        end
```

```
      subroutine CmdK(B,ChrK,E,A)

      Character*1 B,E,A
      Character*4 ChrK

c               'K' command subroutine
c               sends the integer multiplier
c               for continuous data transmission,
c               K, to the Micas.

      character*1 Ackn

      write(3,6020) B,ChrK,E
6020  format('+',A1,'K ',A4,A1)

      read(3,6022) Ackn
6022  format(A1)
      if (Ackn .NE. A) then
           type 6024
6024       format('0','****ERROR: Cmd K not accepted by
     + Micas.****')
      end if

      return
      end



      subroutine CmdP(B,Cp,E,A)

      Character*1 B,E,A
      Character*2 Cp

c               'P' command subroutine indicates
c               to Micas the number of active
c               "fault" data channels .

      Character*1 Ackn

      Write(3,6030) B,Cp,E
6030  format('+',A1,'P ',A2,A1)

      read(3,6032) Ackn
6032  format(A1)

      if (Ackn .NE. A) then
           type 6034
6034       format('0','****ERROR: Cmd P not accepted by
     + Micas.****')
      end if

      return
      end
```

```
        subroutine CmdE(B,CNum,HexAdd,E,A)

        Character*1 B,E,A
        Character*2 CNum
        Character*4 HexAdd

c                      'F' command subroutine defines a
c                      Hex 'fault' data address for
c                      a given channel number.  This
c                      information is sent to the Micas.

        Character*1 Ackn

        Write(3,6040) B,CNum,HexAdd,E
6040    format('+',A1,'E ',A2,' ',A4,A1)

        read(3,6042) Ackn
6042    format(A1)

        if (Ackn .NE. A) then
             type 6044
6044         format('0','****ERROR: Cmd E not accepted by
     + Micas.****')
        end if

        return
        end

        subroutine CmdD(B,Cdly,E,A)

        Character*1 B,E,A
        Character*2 Cdly

c                      'D' command subroutine indicates
c                      to Micas the delay in cycles that
c                      is to occur after a fault condition
c                      before data storage ceases.

        Character*1 Ackn

        Write(3,6030) B,Cdly,E
6030    format('+',A1,'D ',A2,A1)

        read(3,6032) Ackn
6032    format(A1)

        if (Ackn .NE. A) then
             type 6034
6034         format('0','****ERROR: Cmd D not accepted by
     + Micas.****')
        end if

        return
        end
```

```
      subroutine CmdC(B,Ctrl,E)

      character*1 B,E,Ctrl
      integer 1

c               'C' command subroutine sends a control
c               character as follows:
c                       R:  Start Continuous transmission.
c                       S:  Stop Continuous transmission.
c                       D:  Download Fault data.
c                       C:  Reset Fault data acquisition.
c               Note: Acknowledge proceedings are left to
c               the specific situation.

      write(3,4006) B,Ctrl,E
4006  format('+',A1,'C ',A1,A1)

      return
      end
```

```
c       *****   GRAPH   *****   JUNE 1985   *****   R. KNOX

c               This program retrieves data from either
c               'dcos5.ud1' or 'dcos5.ud2' (created by dcos5)
c               and plots the data using information from the
c               'dcos5.prm' (also created by dcos5) and the
c               graphing  subroutine 'Draw'. An optional print
c               out is obtained using subroutine 'Prplot'.


        parameter (N = 1000, M = 5, NC = 4)

c               YScale converts +127 (8 bit max) to 1.0 PU.
        parameter (YScale = 0.0078125)

c               TbaseC is the base sample time for continuous
c               transmission mode. ( = 50 ms )
        parameter (TBaseC = 0.05)

c               TBaseF is the sample time for fault acquisition
c               mode. ( = 20 ms )
        parameter (TBaseF = 0.02)

        Integer IRecL,line,Size,I,IBKS,IX,Num
        Integer NOR,C,NF,In,IK,Ip
        Integer LMY(M),KX(NC),KY(NC),Line(NC)
        Integer InpRec(500)
        Integer Pn,PChan(NC)
        Virtual Y(N,M)
        Logical Fdat,Plot


        Fdat = .false.
        Plot = .false.

c               Get parameters from dcos5 (main program).
        call Parms(NOR,C,NF,In,IK,Ip)

        type 600,NOR,C,NF,In,IK,Ip
600     format(6I3)


c               Obtain plotting information.
        call Spec(In,Ip,Pn,PChan,NC,Fdat,Plot)

        if (Fdat) then
                IX = 250 / (2 * Ip)
                IRecL = IX * Ip
                Num = N / IX
                Size = Num * IX
                IBKS = 1
                do 140 I = 1, Size
                        Y(I,1) = I * TBaseF
140             continue
```

```
                call FDo(InpRec,IRecL,Ip,YScale,Y,N,M,Num,IBKS)
        else
                IRecL = C * NOR * In
                Num = 8
                Size = Num * C * NOR * 1
                IBKS = IRecL / 512 + 1
                do 130 I = 1, Size
                        Y(I,1) = I * IK * TBaseC
130             continue
                call Orgnze(InpRec,IRecL,In,YScale,Y,N,M,Num,
     +                          IBKS)
        end if

        do 50 I = 1, M
                LMY(I) = Size
50      continue

        do 60 I = 1, NC
                KX(I) = 1
                Line(I)=1
60      continue

        do 70 I = 1, Pn
                KY(I) = PChan(I) + 1
70      continue

        do 80 I = Pn+1,NC
                KY(I) = 0
80      continue

        line = 1

c       type 150,(Y(I,2),I=1,Size)
c150    format(F10.4)

        call Draw(Y,N,M,LMY,NC,KY,KX,ierr,Line)

        if (Plot) then
                call Prplot
        end if


        end



        subroutine Orgnze(IGet,IRecL,In,YScale,Y,N,M,Num,IBKS)

        Integer IGet(IRecL),In,IRecL,N,M,Num,IBKS
        Virtual Y(N,M)


        Integer ptr,I,IG,J,L
        open(unit = 4, file = 'dcos5.ud1', status = 'old',
     +       dispose = 'save', access = 'sequential',
     +       organization = 'sequential', blocksize = IBKS,
     +       form = 'unformatted', recl = IRecL,
     +       recordtype = 'variable', initialsize = 1,
     +       extendsize = 1)


        IG = 1
        ptr = 1
        do 100 I = 1, Num
                read (4) IGet
                K = 1
                do 110 J = ptr, (ptr+IRecL-1), In
c               do 110 J = ptr, (ptr+IRecL-1)
                        do 120 L = 1, In
                        Y(IG,(L+1)) = IGet(K) * YScale
c                               Y(J,2) = IGet(K) * YScale
                                K = K + 1
120                     continue
                        IG = IG + 1
110             continue
                ptr = ptr + IRecL
100     continue

        close(unit = 4, dispose = 'save')

        return
        end



        subroutine FDo(IGet,IRecL,Ip,YScale,Y,N,M,Num,IBKS)

        Integer IGet(IRecL),Ip,IRecL,Num,IBKS
        Virtual Y(N,M)


        Integer ptr,I,IG,J,L

        open(unit = 4, file = 'dcos5.ud2', status = 'old',
     +       dispose = 'save', access = 'sequential',
     +       organization = 'sequential', blocksize = IBKS,
     +       form = 'unformatted', recl = IRecL,
     +       recordtype = 'variable', initialsize = 1,
     +       extendsize = 1)


        IG = 1
        ptr = 1
        do 100 I = 1, Num
                read (4) IGet
                K = 1
                do 110 J = ptr, (ptr+IRecL-1), Ip
                        do 120 L = 1, Ip
                                Y(IG,(L+1)) = IGet(K) * YScale
c                               Y(J,2) = IGet(K) * YScale
                                K = K + 1
120                     continue
                        IG = IG + 1
110             continue
                ptr = ptr + IRecL
100     continue

        close(unit = 4, dispose = 'save')

        return
        end



        subroutine Parms(NOR,C,NF,In,IK,Ip)

        Integer NOR,C,NF,In,IK,Ip

        open(unit = 4, file = 'dcos5.prm', status = 'old',
      +      dispose = 'save', access = 'sequential',
      +      organization = 'sequential', blocksize = 1,
      +      form = 'unformatted', recl = 24,
      +      recordtype = 'variable', initialsize = 1,
      +      extendsize = 1)

        read(4) NOR,C,NF,In,IK,Ip

        close(unit = 4, dispose = 'save')

        return
        end



        subroutine Spec(In,Ip,Pn,PChan,NC,Fdat,Plot)

        Integer In,Ip,Pn,PChan(NC)
        Logical Fdat,Plot

        Character*1 Chr,Dtype
        Integer Itest

        type 01
 01     format('1',T20,'DCOS5 DATA GRAPHING ROUTINE'//)

        type 20
 20     format(' ',T5,'Which data are to be graphed?'/
      + T5,'Type C for Continuous data or'/
      + '$',T10,'F for Fault data :  ')
```

```
          accept 21,Dtype
21        format(A1)

          if (Dtype .EQ. 'C') then
                  type 02,In
02                format('0',T5,I2,' continuous channels have
      + been stored.')
                  Fdat = .false.
                  Itest = In
          else
                  type 22,Ip
22                format('0',T5,I2,' fault channels have been
      + stored.')
                  Fdat = .true.
                  Itest = Ip
          end if

          type 23
23        format('$',T5,'How many channels are to be displayed?
      +:  ')

200       accept 03,Pn
03        format(I2)

          if (Pn .GT.  In) then
                  type 04
04                format('0',T5,'Number of channels must be less
      + than number stored:  '/)
                  go to 200
          end if

220       type 05
05        format(' ',T5,'Which channels are to be displayed?'/
      +  T5,'Type "Y" for yes.'/)

          J = 1
          do 210 I = 1, Itest
                  type 06,I
06                format('$',T20,'Channel ',I2,'? ')
                  accept 07,Chr
07                format(A1)
                  if (Chr .EQ. 'Y') then
                          PChan(J) = I
                          Chr = ' '
                          J = J + 1
                  end if
210       continue

          J = J - 1
          if (J .GT. Pn) then
                  type 08
08                format(' ',T5,'Too many channels!'/)
                  go to 220
          end if
```

```
        type 24
24      format('0',T5,'Send resulting plot to printer? (Y or
     + N) ',$)

        accept 25,Chr
25      format(A1)

        if (Chr .EQ. 'Y') then
                Plot = .true.
        end if

        type 09
09      format('1',T35,'wait')


        return
        end
```

Appendix A6

MICAS - PDP11 Communications Systems

1 AUX1 (CIOINIT)

2 AUX2 (CIOBGR)

3 SERE (CIOINT_V5) for MICAS

4 CODA (CIOSYS)

5 FADA (CIOFDS)

6 DCOS5 for PDP11

```
FILE: CIOINIT:pRBK      HEWLETT-PACKARD: 8086 Assembler                         Wed, 12 Jun 1985, 15:24     PAGE   1

LOCATION OBJECT CODE LINE     SOURCE LINE

                         1         "8086"
                         2         NAME "CIOINIT CALL ADDR-21C6"
                         3
                         4 *       MICAS COMMUNICATIONS INITIALIZATION BLOCK
                         5 *
                         6 *       The auxillary data communications port is
                         7 *       initialiazed and a heading is sent to the
                         8 *       terminal.
                         9
                        10         PROG
                        11
              <0084>    12 SBASE   EQU     0084H            MAIN SERIAL PORT BASE ADDR
              <0000>    13 SERIO   EQU     0000H            MAIN IO PORT
              <0002>    14 SERSTA  EQU     0002H            MAIN CMD/STATUS REGISTER
              <A102>    15 AUXSTA  EQU     0A102H           AUX SERIAL PORT CMD/STATUS
              <0004>    16 EOT     EQU     04H              END OF TRANSMISSION
                        17
   0000 51              18         PUSH    CX
   0001 B90100          19         MOV     CX,#1H        INITIAL DELAY
   0004 B90100          20         MOV     CX,#1H
   0007 E2FE            21 DELAY   LOOP    DELAY
   0009 B98400          22         MOV     CX,#SBASE
   000C E81F00          23         CALL    TITLE             PRINT IDENTIFIER.
                        24
   000F C60602A1AD      25         MOV BYTE PTR[AUXSTA],#0ADH   ENSURE THAT DEVICE
   0014 C60602A1AD      26         MOV BYTE PTR[AUXSTA],#0ADH   IS IN COMMAND MODE.
   0019 C60602A1FF      27         MOV BYTE PTR[AUXSTA],#0FFH   PERFORM INTERNAL RESET.
   001E C60602A16E      28         MOV BYTE PTR[AUXSTA],#6EH    1 STOP BIT,NO PAR, DL 8,16X
   0023 C60602A136      29         MOV BYTE PTR[AUXSTA],#36H    RCVR ON,XMTR OFF, CLEAR ERRORS
   0028 83C604          30         ADD     SI,#04H
   002B 59              31         POP     CX
                        32
   002C FF24            33         JMP WORD PTR[SI]
                        34
                        35
                        36
                        37 *** OUTPUT THE HEADING " MCOS VER 4.0" ***
                        38
   002E BB003C          39 TITLE   MOV     BX,#HDING
   0031 8A07            40 LOOPX   MOV     AL,[BX]
   0033 E81D00          41         CALL    PUTCHAR
   0036 43              42         INC     BX
   0037 3C04            43         CMP     AL,#EOT
   0039 75F6            44         JNE     LOOPX
   003B C3              45         RET
                        46
   003C 0D0A            47 HDING   DB      0DH,0AH
   003E 2020202020      48         ASCII   "      MCOS VER 4.0"
   0050 0D0A04          49         DB      0DH,0AH,04H
                        50
                        51
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      52
                      53 ### SEND A CHARACTER TO THE MAIN (MONITOR) SERIAL PORT ###
                      54
    0053 1E           55 PUTCHAR  PUSH      DS
    0054 8ED9         56          MOV       DS,CX
    0056 8A260200     57 WAITO    MOV       AH,DS:[SERSTA]


    005A 80E401       58          AND       AH,#01H
    005D 74F7         59          JZ        WAITO
    005F A20000       60          MOV       DS:[SERIO],AL
    0062 1F           61          POP       DS
    0063 C3           62          RET

Errors=    0
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1          "8086"
                      2          NAME "CIOBGR CALL ADDR-21CC"
                      3
                      4 !        MICAS COMMUNICATIONS BACKGROUND MONITOR
                      5 !
                      6 !        This is a cyclical routine that runs with
                      7 !        a very low priority (i.e. large cycle time).
                      8 !        In this sense it is essentially a background
                      9 !        job.  The purpose of this routine is to monitor
                     10 !        the receiver buffer for incoming characters from
                     11 !        the remote system.  Upon detecting a non-empty
                     12 !        buffer, the character string is parsed and where
                     13 !        valid commands are found the appropriate action is
                     14 !        taken and/or the vital information if decoded.
                     15
                     16          PROG
                     17
<0002>               18 STX      EQU       02H                  START TEXT PROTOCOL
<0003>               19 ETX      EQU       03H                  END TEXT PROTOCOL
<0006>               20 ACK      EQU       06H                  ACKNOWLEDGE PROTOCOL
<0020>               21 SPACE    EQU       20H                  BLANK CHAR.
                     22
                     23
                     24
                     25 !!!!  CIOBGR PARAMETER LIST (PASSED VARS AND CONSTANTS) !!!!
                     26
<0000>               27 S_CA       EQU        00H         call addr
<0002>               28 S_AK       EQU        02H         ACS
<0004>               29 S_RS       EQU        04H         sys R/S flag
<0006>               30 S_K        EQU        06H         K
<0008>               31 S_N        EQU        08H         n
<000A>               32 S_ND       EQU        0AH         ND
<000C>               33 S_DY       EQU        0CH         DLY
<000E>               34 S_FP       EQU        0EH         FSTOP
<0010>               35 S_FT       EQU        10H         FHALT
<0012>               36 S_FQ       EQU        12H         Freq
<0014>               37 S_CH       EQU        14H         data channel base
                     38
                     39
                     40
                     41 !!!!   SYSTEM R MAP (INIT FLAGS/RCVR BUFFER AND POINTERS) !!!!
                     42
<0000>               43 R_X        EQU        00H         pointer to X map
<0002>               44 R_F        EQU        02H         pointer to F map
<0004>               45 R_NX       EQU        04H         Nmax
<0006>               46 R_PX       EQU        06H         Pmax
<0008>               47 R_IT       EQU        08H         INIT flag
<000A>               48 R_FT       EQU        0AH         FINIT flag
<000C>               49 R_BR       EQU        0CH         BLR
<000E>               50 R_IR       EQU        0EH         IPR
<0010>               51 R_OR       EQU        10H         OPR
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

          <0012>      52 R_RB        EQU         12H         rcvr data base
                      53
                      54
                      55
   0000 52            56             PUSH       DX
                      57


   0001 8B5D10        58 BEGIN       MOV        BX,[DI+R_OR]        GET OPR
   0004 3B5D0E        59             CMP        BX,[DI+R_IR]        OPR = IPR ?
   0007 7448          60             JE         EXIT1               NO CHARS IN BUFFER
                      61
   0009 3B5D0C        62             CMP        BX,[DI+R_BR]        OPR >= BLR ?
   000C 7C02          63             JL         SKIPR
   000E 33DB          64             XOR        BX,BX
   0010 8A4112        65 SKIPR       MOV        AL,[BX][DI+R_RB]
   0013 43            66             INC        BX                  INC OPR
   0014 895D10        67             MOV        [DI+R_OR],BX
   0017 3C02          68             CMP        AL,#STX
   0019 7536          69             JNE        EXIT1
                      70
   001B E8F001        71 T1CMD1      CALL       GETBCHR
   001E 8AC8          72             MOV        CL,AL               STORE FIRST CHAR
   0020 3C4E          73             CMP        AL,#4EH             = 'N'
   0022 7503          74             JNE        T2CMD1
                      75
   0024 E92D00        76             JMP        CMD1
                      77
   0027 3C50          78 T2CMD1      CMP        AL,#50H             = 'P'
   0029 7503          79             JNE        T1CMD2
                      80
   002B E92600        81             JMP        CMD1
                      82
   002E 3C41          83 T1CMD2      CMP        AL,#41H             = 'A'
   0030 7503          84             JNE        T2CMD2
                      85
   0032 E96500        86             JMP        CMD2
                      87
   0035 3C45          88 T2CMD2      CMP        AL,#45H             = 'E'
   0037 7503          89             JNE        T1CMD3
                      90
   0039 E95E00        91             JMP        CMD2
                      92
   003C 3C4B          93 T1CMD3      CMP        AL,#4BH             = 'K'
   003E 7503          94             JNE        T2CMD3
                      95
   0040 E9C900        96             JMP        CMD3
                      97
   0043 3C44          98 T2CMD3      CMP        AL,#44H             = 'D'
   0045 7503          99             JNE        T1CMD4
                     100
   0047 E9FB00       101             JMP        CMD4
                     102
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   00ED 8AE0          205           MOV       AH,AL
   00EF E81C01        206           CALL      GETBCHR             LOWER ASCII CHAR, 2ND BYTE
   00F2 E8F500        207           CALL      TOHEX               FORM SINGLE LOWER BYTE
   00F5 8AE6          208           MOV       AH,DH               16 BIT ADDR NOW FORMED
   00F7 50            209           PUSH      AX                  future data addr checks
   00F8 E81301        210           CALL      GETBCHR
   00FB 3C03          211           CMP       AL,#ETX             PROTOCOL COMPLETED?
   00FD 58            212           POP       AX
   00FE 7509          213           JNE       EXIT3
                      214
   0100 8907          215           MOV       [BX],AX             STORE NODE ADDR
   0102 B006          216           MOV       AL,#ACK             ACKNOWLEDGE COMPLETED
   0104 8B5C02        217           MOV       BX,[SI+S_AK]        COMMAND
   0107 8807          218           MOV       [BX],AL
   0109 E9C400        219 EXIT3     JMP       ENDBLK
                      220
                      221
                      222
                      223 ***       COMMAND 'K' IS FOLLOWED BY
                      224 *         A 16 BIT NO. WHICH PROVIDES INTEGER
                      225 ***       MULTIPLES OF THE SAMPLING PERIOD.
                      226
   010C E8FF00        227 CMD3      CALL      GETBCHR
   010F 3C20          228           CMP       AL,#SPACE


   0111 752F          229           JNE       EXIT4
                      230
   0113 E8F800        231           CALL      GETBCHR             UPPER ASCII CHAR.
   0116 8AE0          232           MOV       AH,AL
   0118 E8F300        233           CALL      GETBCHR             LOWER ASCII CHAR.
   011B E8CC00        234           CALL      TOHEX               FORM UPPER HEX BYTE
   011E 8AF0          235           MOV       DH,AL
   0120 E8EB00        236           CALL      GETBCHR             UPPER ASCII CHAR.
   0123 8AE0          237           MOV       AH,AL
   0125 E8E600        238           CALL      GETBCHR             LOWER ASCII CHAR.
   0128 E8BF00        239           CALL      TOHEX               FORM LOWER HEX BYTE
   012B 8AE6          240           MOV       AH,DH
   012D 50            241           PUSH      AX
   012E E8DD00        242           CALL      GETBCHR
   0131 3C03          243           CMP       AL,#ETX             PROTOCOL COMPLETED?
   0133 58            244           POP       AX
   0134 750C          245           JNE       EXIT4
                      246
   0136 8B5C06        247           MOV       BX,[SI+S_K]         GET EA
   0139 8907          248           MOV       [BX],AX             STORE
   013B B006          249           MOV       AL,#ACK
   013D 8B5C02        250           MOV       BX,[SI+S_AK]
   0140 8807          251           MOV       [BX],AL             ACKNOWLEDGE COMPLETED
                      252                                         ;COMMAND
   0142 E98800        253 EXIT4     JMP       ENDBLK
                      254
                      255
```

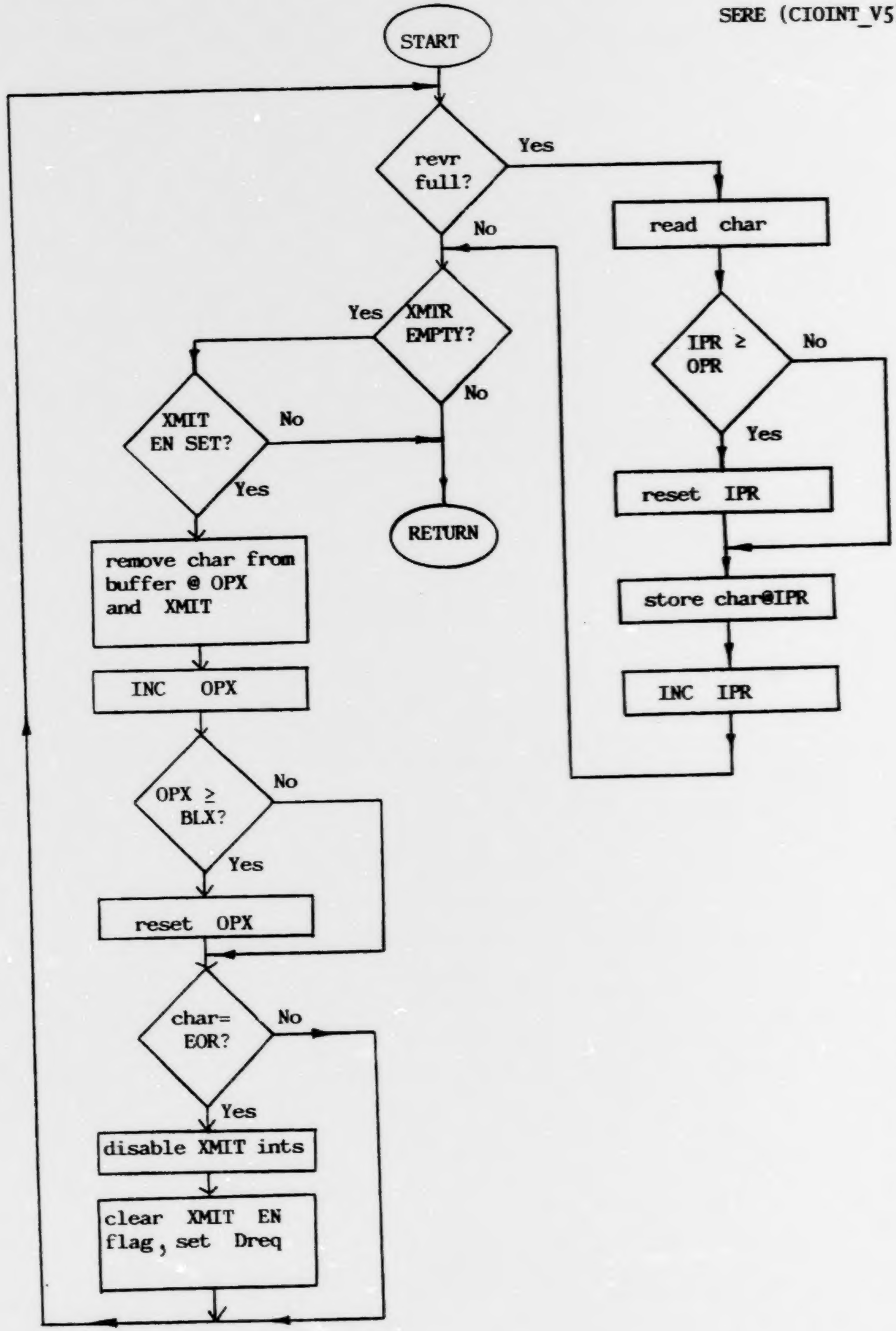
```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     256
                     257 ***       Command 'D' specifies an 8 bit number representing
                     258 *         the delay, in cycles, which is to occur after a
                     259 ***       fault condition before data acquisition ceases.
                     260
   0145 E8C600       261 CMD4      CALL      GETBCHR
   0148 3C20         262           CMP       AL,#SPACE
   014A 7527         263           JNE       EXIT5
                     264
   014C E8BF00       265           CALL      GETBCHR             UPPER CHAR
   014F 8AE0         266           MOV       AH,AL
   0151 E8BA00       267           CALL      GETBCHR             LOWER CHAR
   0154 50           268           PUSH      AX
   0155 E8B600       269           CALL      GETBCHR
   0158 3C03         270           CMP       AL,#ETX             PROTOCOL COMPLETED?
   015A 58           271           POP       AX
   015B 7516         272           JNE       EXIT5
                     273
   015D E8BA00       274           CALL      TOHEX
   0160 32E4         275           XOR       AH,AH
   0162 8B5C0C       276           MOV       BX,[SI+S_DY]
   0165 8907         277           MOV       [BX],AX             STORE DELAY
   0167 C7450A0000   278           MOV WORD PTR[DI+R_FT],#0000H  CLEAR FINIT FLAG
   016C B006         279           MOV       AL,#ACK
   016E 8B5C02       280           MOV       BX,[SI+S_AK]        ACKNOWLEDGE
   0171 8807         281           MOV       [BX],AL
                     282
   0173 E95A00       283 EXIT5     JMP       ENDBLK
                     284
                     285 -


                     286
                     287 ***       Command 'C' controls the data acquisition
                     288 *         system by setting and clearing certain flags
                     289 *         as follows:
                     290 *             'R' sets the system R/S flag which starts
                     291 *                 the data base 1 acquisition and transmission.
                     292 *             'S' clears the system R/S flag thus stopping
                     293 *                 data base 1 acquisition and transmission.
                     294 *             'D' sets the Freq flag which requests a
                     295 *                 data base 2 memory dump.
                     296 *             'C' clears the FSTOP and FHALT flags putting
                     297 *                 the data base 2 system into its normal
                     298 ***               acquisition mode.
                     299
   0176 E89500       300 CMD5      CALL      GETBCHR
   0179 3C20         301           CMP       AL,#SPACE
   017B 7553         302           JNE       ENDBLK
                     303
   017D E88E00       304           CALL      GETBCHR
   0180 8AE0         305           MOV       AH,AL
   0182 E88900       306           CALL      GETBCHR
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   0185 3C03          307            CMP       AL,#ETX
   0187 7547          308            JNE       ENDBLK
                      309
   0189 80FC52        310            CMP       AH,#52H            = 'R'
   018C 750A          311            JNE       REQ2
                      312
   018E 8B5C04        313            MOV       BX,[SI+S_RS]
   0191 C707FFFF      314            MOV WORD PTR[BX],#0FFFFH       SET R/S FLAG
   0195 E93800        315            JMP       ENDBLK
                      316
   0198 80FC53        317 REQ2       CMP       AH,#53H            = 'S'
   019B 750A          318            JNE       REQ3
                      319
   019D 8B5C04        320            MOV       BX,[SI+S_RS]
   01A0 C7070000      321            MOV WORD PTR[BX],#0000H        CLEAR R/S FLAG
   01A4 E92200        322            JMP       RESPOND
                      323
   01A7 80FC44        324 REQ3       CMP       AH,#44H            = 'D'
   01AA 750A          325            JNE       REQ4
                      326
   01AC 8B5C12        327            MOV       BX,[SI+S_FQ]
   01AF C707FFFF      328            MOV WORD PTR[BX],#0FFFFH       SET Freq FLAG
   01B3 E91A00        329            JMP       ENDBLK
                      330
   01B6 80FC43        331 REQ4       CMP       AH,#43H            = 'C'
   01B9 7515          332            JNE       ENDBLK
                      333
   01BB 8B5C0E        334            MOV       BX,[SI+S_FP]
   01BE C7070000      335            MOV WORD PTR[BX],#0000H        CLEAR FSTOP FLAG
   01C2 8B5C10        336            MOV       BX,[SI+S_FT]
   01C5 C7070000      337            MOV WORD PTR[BX],#0000H        CLEAR FHALT FLAG
                      338
   01C9 B006          339 RESPOND    MOV       AL,#ACK
   01CB 8B5C02        340            MOV       BX,[SI+S_AK]       ACKNOWLEDGE COMPLETED
   01CE 8807          341            MOV       [BX],AL            COMMAND
                      342


   01D0 8B4504        343 ENDBLK     MOV       AX,[DI+R_NX]
   01D3 D1E0          344            SAL       AX,1               PTR TO NEXT BLOCK =
   01D5 03F0          345            ADD       SI,AX              SI + 16H + 2 * Nmax
   01D7 8B4506        346            MOV       AX,[DI+R_PX]                + 2 * Pmax
   01DA D1E0          347            SAL       AX,1
   01DC 03F0          348            ADD       SI,AX
   01DE 83C616        349            ADD       SI,#16H
                      350
   01E1 037D0C        351            ADD       DI,[DI+R_BR]       DI' = BLR + R_RB
   01E4 83C712        352            ADD       DI,#R_RB
                      353
   01E7 5A            354            POP       DX
                      355
   01E8 FF24          356            JMP WORD PTR[SI]
                      357
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                     358
                     359
                     360 ***      TOHEX TAKES TWO ASCII CHARACTERS REPRESENTING
                     361 *        UPPER AND LOWER NIBBLES OF A HEX BYTE, AND
                     362 ***      CONVERTS THEM INTO A SINGLE HEX BYTE.
                     363
   01EA 80FC40       364 TOHEX    CMP      AH,#40H
   01ED 7706         365          JA       HEX1
   01EF 80EC30       366          SUB      AH,#30H
   01F2 E90300       367          JMP      NEXT
   01F5 80EC37       368 HEX1     SUB      AH,#37H
   01F8 D0E4         369 NEXT     SHL      AH,1
   01FA D0E4         370          SHL      AH,1
   01FC D0E4         371          SHL      AH,1
   01FE D0E4         372          SHL      AH,1
   0200 3C40         373          CMP      AL,#40H
   0202 7705         374          JA       HEX2
   0204 2C30         375          SUB      AL,#30H
   0206 E90200       376          JMP      CMB
   0209 2C37         377 HEX2     SUB      AL,#37H
   020B 02C4         378 CMB      ADD      AL,AH
   020D C3           379          RET
                     380
                     381
                     382
                     383 *** THIS ROUTINE GETS A CHARACTER FROM
                     384 *   THE RECEIVER BUFFER AND RETURNS
                     385 *** IT IN AL.
                     386
   020E 53           387 GETBCHR  PUSH     BX
   020F 8B5D10       388 WAITB    MOV      BX,[DI+R_OR]                  GET OPR
   0212 3B5D0E       389          CMP      BX,[DI+R_IR]                  OPR = IPR ?
   0215 74F8         390          JE       WAITB
   0217 3B5D0C       391          CMP      BX,[DI+R_BR]                  OPR >= BLR ?
   021A 7C02         392          JL       SKIPG
   021C 33DB         393          XOR      BX,BX
   021E 8A4112       394 SKIPG    MOV      AL,[BX][DI+R_RB]
   0221 43           395          INC      BX                            INC OPR
   0222 895D10       396          MOV      [DI+R_OR],BX
   0225 5B           397          POP      BX
   0226 C3           398          RET
                     399


                     400

Errors=    0
```

Program:

SERE (CIOINT_V5)


START
revr full?
Yes
No
read char
XMTR EMPTY?
Yes
No
IPR ≥ OPR
No
Yes
XMIT EN SET?
No
Yes
reset IPR
RETURN
remove char from buffer @ OPX and XMIT
store char@IPR
INC OPX
INC IPR
OPX ≥ BLX?
No
Yes
reset OPX
char= EOR?
No
Yes
disable XMIT ints
clear XMIT EN flag, set Dreq

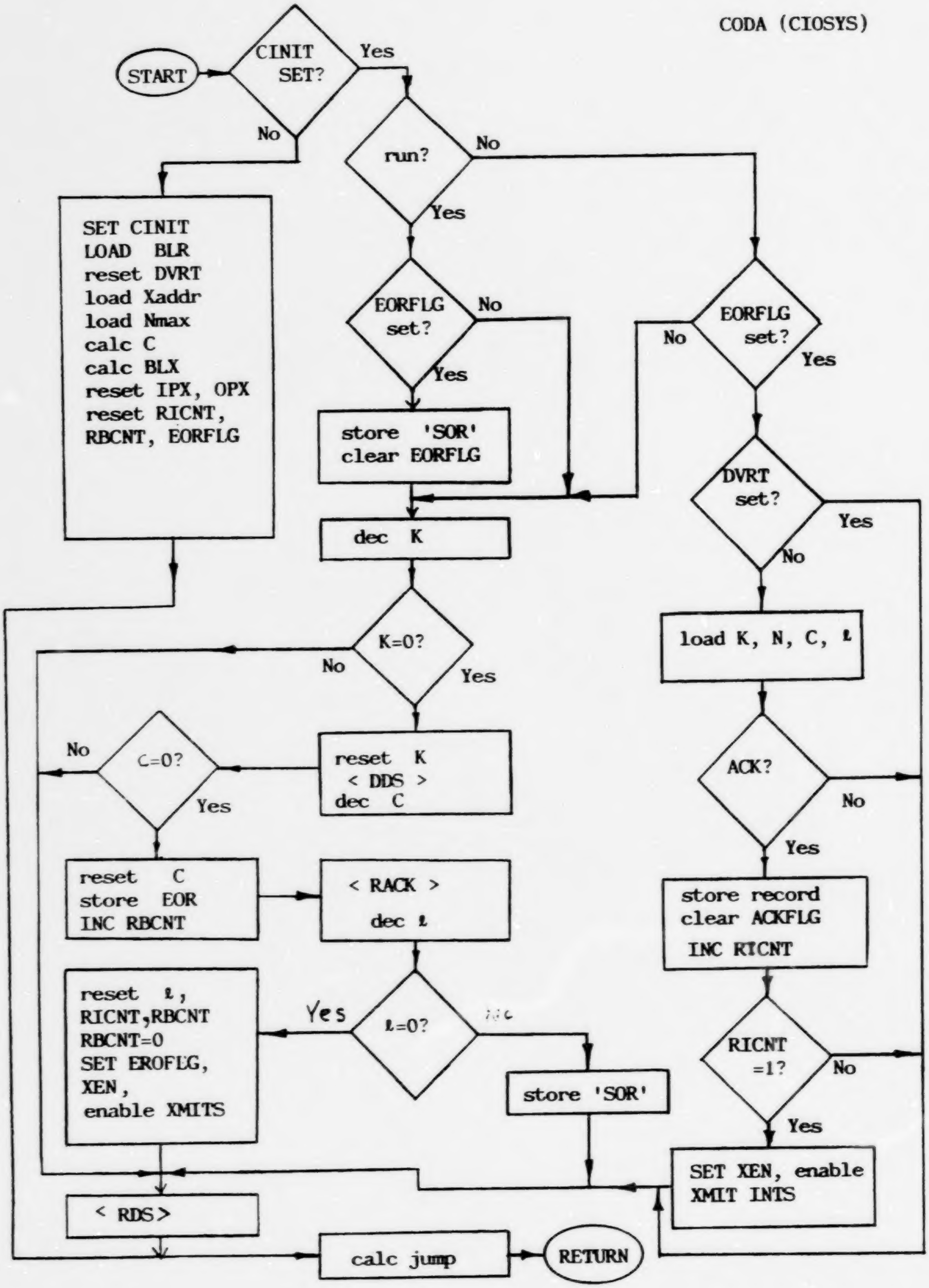
```
LOCATION OBJECT CODE LINE     SOURCE LINE

                       1          "8086"
                       2
                       3          NAME  "CIOINT_V5 ADDR - 21C9"
                       4
                       5 !        MICAS AUXILLARY COMMUNICATIONS I/O INT. BLOCK,VER 5
                       6 !
                       7 !        This interrupt routine is event triggered by the
                       8 !        interrupt request pin on the auxillary serial port.
                       9 !        The routine detects whether a transmit interrupt, a
                      10 !        receiver interrupt, or both have occurred.  For a
                      11 !        transmit interrupt, a character is removed from the
                      12 !        system xmit buffer and sent to the serial port.  For
                      13 !        a receive interrupt, a character is read from the
                      14 !        serial port and stored in the system rcvr buffer.
                      15 !        Synchronization parameters are set as well during
                      16 !        a transmit interrupt.
                      17
                      18          PROG
                      19
      <A100>          20 AUXIO    EQU      0A100H              AUX SERIAL I/O PORT
      <A102>          21 AUXSTA   EQU      0A102H              AUX SERIAL CMD/STATUS REG
      <000D>          22 EOR      EQU      0DH                 END OF RECORD = CR
                      23
                      24
                      25
                      26 !!!!  CIOINT_V5 PARAMETER LIST (PASSED VARS AND CONSTANTS)  !!!!
                      27
      <0000>          28 S_CA        EQU         00H         call address
      <0002>          29 S_DQ        EQU         02H         Dreq
      <0004>          30 S_XN        EQU         04H         XEN
                      31
                      32
                      33
                      34 !!!!   SYSTEM R MAP (INIT FLAGS/RCVR BUFFER AND POINTERS) !!!!
                      35
      <0000>          36 R_X         EQU         00H         pointer to X map
      <0002>          37 R_F         EQU         02H         pointer to F map
      <0004>          38 R_NX        EQU         04H         Nmax
      <0006>          39 R_PX        EQU         06H         Pmax
      <0008>          40 R_IT        EQU         08H         INIT flag
      <000A>          41 R_FT        EQU         0AH         FINIT flag
      <000C>          42 R_BR        EQU         0CH         BLR
      <000E>          43 R_IR        EQU         0EH         IPR
      <0010>          44 R_OR        EQU         10H         OPR
      <0012>          45 R_RB        EQU         12H         rcvr data base
                      46
                      47
                      48
                      49 !!!!   SYSTEM X MAP (XMTR POINTER AND BUFFER)  !!!!
                      50
      <0000>          51 X_R         EQU         00H         pointer to R map
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

          <0002>      52 X_CO       EQU        02H        C#
          <0004>      53 X_BX       EQU        04H        BLX
          <0006>      54 X_IX       EQU        06H        IPX
          <0008>      55 X_OX       EQU        08H        OPX
          <000A>      56 X_K        EQU        0AH        K
          <000C>      57 X_N        EQU        0CH        n


          <000E>      58 X_C        EQU        0EH        C
          <0010>      59 X_L        EQU        10H        l
          <0012>      60 X_RI       EQU        12H        RICNT
          <0014>      61 X_RB       EQU        14H        RBCNT
          <0016>      62 X_DT       EQU        16H        DCNT
          <0018>      63 X_EG       EQU        18H        EORFLG
          <001A>      64 X_XB       EQU        1AH        xmtr data base
                      65
                      66
                      67
0000 A002A1           68 TSTR      MOV        AL,[AUXSTA]
0003 2402             69           AND        AL,#02H             RECEIVED A CHAR ?
0005 750A             70           JNZ        RCVR
                      71
0007 A002A1           72 TSTX      MOV        AL,[AUXSTA]
000A 2401             73           AND        AL,#01H             READY TO XMIT ?
000C 7519             74           JNZ        XMTR
                      75
000E E94F00           76           JMP        IFINI
                      77
0011 A000A1           78 RCVR      MOV        AL,[AUXIO]          GET CHARACTER
0014 8B5D0E           79           MOV        BX,[DI+R_IR]        GET IPR
0017 3B5D0C           80           CMP        BX,[DI+R_BR]        IPR >= BLR
001A 7C02             81           JL         ISKIP1
                      82
001C 33DB             83           XOR        BX,BX               IF SO, RESET
001E 884112           84 ISKIP1    MOV        [BX][DI+R_RB],AL    STORE CHAR. IN CMD BUFFER
0021 43               85           INC        BX
0022 895D0E           86           MOV        [DI+R_IR],BX        UPDATE IPR
0025 EBE0             87           JMP        TSTX                CHECK TRANSMIT INT
                      88
0027 8B5C04           89 XMTR      MOV        BX,[SI+S_XN]        GET XEN FLAG
002A 833F00           90           CMP WORD PTR[BX],#0000H        FLAG SET?
002D 7431             91           JE         IFINI               NO
                      92
002F 8B3D             93           MOV        DI,[DI]             POINT TO XMIT BUFFER
0031 8B5D08           94           MOV        BX,[DI+X_OX]        GET OPX
0034 8A411A           95           MOV        AL,[BX][DI+X_XB]    GET CHAR FROM BUFFER
0037 A200A1           96           MOV        [AUXIO],AL          XMIT CHAR
003A 43               97           INC        BX
003B 3B5D04           98           CMP        BX,[DI+X_BX]        OPX >= BLX
003E 7C02             99           JL         ISKIP2
                     100
0040 33DB            101           XOR        BX,BX               IF SO. RESET OPX
0042 895D08          102 ISKIP2    MOV        [DI+X_OX],BX        UPDATE OPX
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0045 3C0D          103          CMP       AL,#EOR
    0047 7513          104          JNE       ISKIP3
                       105
    0049 C60602A126    106          MOV BYTE PTR [AUXSTA],#26H        DISABLE XMIT INT
    004E 8B5C04        107          MOV       BX,[SI+S_XN]
    0051 C7070000      108          MOV WORD PTR[BX],#0000H           CLEAR XEN
    0055 8B5C02        109          MOV       BX,[SI+S_DQ]
    0058 C707FFFF      110          MOV WORD PTR[BX],#0FFFFH          SET DREQ
                       111
    005C 8B3D          112 ISKIP3   MOV       DI,[DI]            POINT TO CMD BUFFER
    005E EBA0          113          JMP       TSTR               CHECK RCVR INT
                       114


    0060 037D0C        115 IFINI    ADD       DI,[DI+R_BR]       CALC NEW DI
    0063 83C712        116          ADD       DI,#R_RB
    0066 83C608        117          ADD       SI,#08H            CALC NEW SI
                       118
    0069 FF24          119          JMP WORD PTR[SI]

Errors=    0
```

Program:

CODA (CIOSYS)


START
CINIT SET?
Yes
No
SET CINIT
LOAD BLR
reset DVRT
load Xaddr
load Nmax
calc C
calc BLX
reset IPX, OPX
reset RICNT,
RBCNT, EORFLG
run?
No
Yes
EORFLG set?
No
Yes
store 'SOR'
clear EORFLG
dec K
K=0?
No
Yes
reset K
< DDS >
dec C
C=0?
No
Yes
reset C
store EOR
INC RBCNT
< RACK >
dec ℓ
ℓ=0?
Yes
No
reset ℓ,
RICNT,RBCNT
RBCNT=0
SET EROFLG,
XEN,
enable XMITS
store 'SOR'
< RDS >
calc jump
RETURN
EORFLG set?
No
Yes
DVRT set?
Yes
No
load K, N, C, ℓ
ACK?
No
Yes
store record
clear ACKFLG
INC RICNT
RICNT =1?
No
Yes
SET XEN, enable
XMIT INTS

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                        1          "8086"
                        2          NAME "CIOSYS CALL ADDR-21CF"
                        3
                        4 *        MICAS CONTINUOUS DATA TRANSMISSION BLOCK VER 3.0
                        5 *
                        6 *        This block may be considered the main operating
                        7 *        block of the MCOS3 system.  Its functions are
                        8 *        as follows:
                        9 *        1) to initialize and execute the continuous data
                       10 *           processing mode.  This includes acquiring
                       11 *           the requested data, processing it for transmission,
                       12 *           and storing it in a transmission buffer.
                       13 *
                       14 *        2) to handle all asynchronous transmission events.
                       15 *           These take the form of acknowledging received
                       16 *           commands and sending a fault detection protocol.
                       17 *
                       18 *        3) to control the synchronized process of sending
                       19 *           portions of each filled transmit buffer of data
                       20 *           using an interrupt driven technique.  This must
                       21 *           be done for both the continuous data buffer and
                       22 *           the fault data buffer.
                       23
                       24          GLOBAL TOASC
                       25          GLOBAL SADJUST
                       26
                       27          PROG
                       28
    <A100>             29 AUXIO    EQU        0A100H     AUX SERIAL I/O PORT
    <A102>             30 AUXSTA   EQU        0A102H     AUX SERIAL CMD/STATUS REG
    <0006>             31 ACKPRO   EQU        06H        ACKNOWLEDGE PROTOCOL
    <0053>             32 SPRO     EQU        53H        "S" DATA START PROTOCOL
    <000D>             33 EOR      EQU        0DH        END OF RECORD PROTOCOL
                       34
                       35
                       36
                       37 ****  CIOSYS PARAMETER LIST (PASSED VARS AND CONSTANTS) ****
                       38
    <0000>             39 S_CA     EQU        00H       call addr
    <0002>             40 S_BR     EQU        02H       BLR
    <0004>             41 S_B      EQU        04H       b
    <0006>             42 S_NX     EQU        06H       Nmax
    <0008>             43 S_L      EQU        08H       l*
    <000A>             44 S_DT     EQU        0AH       DCNT*
    <000C>             45 S_XR     EQU        0CH       Xaddr*
    <000E>             46 S_DQ     EQU        0EH       Dreq flag
    <0010>             47 S_XN     EQU        10H       XEN  flag
    <0012>             48 S_AK     EQU        12H       ASC  flag
    <0014>             49 S_RS     EQU        14H       SYS R/S flag
    <0016>             50 S_K      EQU        16H       K
    <0018>             51 S_N      EQU        18H       n
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

          <001A>     52 S_DV       EQU        1AH        DVRT flag
          <001C>     53 S_CH       EQU        1CH        Channel base pointer
                     54
                     55
                     56
                     57 ****   SYSTEM R MAP (INIT FLAGS/RCVR BUFFER AND POINTERS) ****


                     58
          <0000>     59 R_X        EQU        00H        pointer to X map
          <0002>     60 R_F        EQU        02H        pointer to F map
          <0004>     61 R_NX       EQU        04H        Nmax
          <0006>     62 R_PX       EQU        06H        Pmax
          <0008>     63 R_IT       EQU        08H        INIT flag
          <000A>     64 R_FT       EQU        0AH        FINIT flag
          <000C>     65 R_BR       EQU        0CH        BLR
          <000E>     66 R_IR       EQU        0EH        IPR
          <0010>     67 R_OR       EQU        10H        OPR
          <0012>     68 R_RB       EQU        12H        rcvr data base
                     69
                     70
                     71
                     72 ****   SYSTEM X MAP (XMTR POINTER AND BUFFER)  ****
                     73
          <0000>     74 X_R        EQU        00H        pointer to R map
          <0002>     75 X_CO       EQU        02H        C#
          <0004>     76 X_BX       EQU        04H        BLX
          <0006>     77 X_IX       EQU        06H        IPX
          <0008>     78 X_OX       EQU        08H        OPX
          <000A>     79 X_K        EQU        0AH        K
          <000C>     80 X_N        EQU        0CH        n
          <000E>     81 X_C        EQU        0EH        C
          <0010>     82 X_L        EQU        10H        I
          <0012>     83 X_RI       EQU        12H        RICNT
          <0014>     84 X_RB       EQU        14H        RBCNT
          <0016>     85 X_DT       EQU        16H        DCNT
          <0018>     86 X_EG       EQU        18H        EORFLG
          <001A>     87 X_XB       EQU        1AH        xmtr data base
                     88
0000 837D0800        89            CMP WORD PTR[DI+R_IT],#0000H        FIRST PASS?
0004 7506            90            JNE        NOFRST                   NO
                     91
0006 E8D800          92            CALL       CINIT                    YES
0009 E9AB00          93            JMP        CALCJ
                     94
000C 8B3D            95 NOFRST     MOV        DI,[DI]                  POINT TO XMIT BUFFER
000E 8B5C14          96            MOV        BX,[SI+S_RS]
0011 833F00          97            CMP WORD PTR[BX],#0000H             TEST R/S FLAG
0014 7511            98            JNE        STEST                    RUN
                     99
0016 837D1800       100            CMP WORD PTR[DI+X_EG],#0000H        STOP,EORFLG SET?
001A 741B           101            JE         CONTIN                   NO
                    102
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    001C 8B5C1A       103             MOV          BX,[SI+S_DV]
    001F 833F00       104             CMP WORD PTR[BX],#0000H            DVRT SET?
    0022 7467         105             JE           RELOAD                NO
                      106
    0024 E98D00       107             JMP          PH2                   YES
                      108
    0027 837D1800     109 STEST       CMP WORD PTR[DI+X_EG],#0000H       EORFLG SET?
    002B 740A         110             JE           CONTIN                NO
                      111
    002D B453         112             MOV          AH,#SPRO              YES,STORE SOR
    002F E89B00       113             CALL         PUTXCHR
    0032 C745180000   114             MOV WORD PTR[DI+X_EG],#0000H       CLEAR EORFLG


                      115
    0037 FF4D0A       116 CONTIN      DEC WORD PTR[DI+X_K]               DEC K
    003A 7578         117             JNZ          PH2
                      118
    003C 8B5C16       119             MOV          BX,[SI+S_K]           RESET K
    003F 8B07         120             MOV          AX,[BX]
    0041 89450A       121             MOV          [DI+X_K],AX
                      122
    0044 33DB         123             XOR          BX,BX
    0046 E8FF00       124             CALL         DDS                   GET, PROCESS DATA SET
                      125
    0049 FF4D0E       126             DEC WORD PTR[DI+X_C]               DEC C
    004C 7566         127             JNZ          PH2
                      128
    004E 8B4502       129             MOV          AX,[DI+X_CO]          RESET C
    0051 89450E       130             MOV          [DI+X_C],AX
                      131
    0054 B40D         132             MOV          AH,#EOR               SEND EOR
    0056 E87400       133             CALL         PUTXCHR
                      134
    0059 FF4514       135             INC WORD PTR[DI+X_RB]              INC RBCNT
    005C E85901       136             CALL         RACK                  RUN MODE ACK HANDLER
                      137
    005F FF4D10       138             DEC WORD PTR[DI+X_L]               DEC L
    0062 754B         139             JNZ          SNR
                      140
    0064 8B5C08       141             MOV          BX,[SI+S_L]           RESET L
    0067 8B07         142             MOV          AX,[BX]
    0069 894510       143             MOV          [DI+X_L],AX
    006C C74518FFFF   144             MOV WORD PTR[DI+X_EG],#0FFFFH      SET EORFLG
    0071 8B4514       145             MOV          AX,[DI+X_RB]          RICNT <== RBCNT
    0074 894512       146             MOV          [DI+X_RI],AX
    0077 C745140000   147             MOV WORD PTR[DI+X_RB],#0000H       RESET RBCNT
    007C 8B5C10       148             MOV          BX,[SI+S_XN]          SET XEN FLAG
    007F C707FFFF     149             MOV WORD PTR[BX],#0FFFFH
    0083 C60602A127   150             MOV BYTE PTR[AUXSTA],#27H          ENABLE XMIT INTS
    0088 E92900       151             JMP          PH2
                      152
    008B 8B5C16       153 RELOAD      MOV          BX,[SI+S_K]           LOAD K
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    00BE 8B07          154             MOV       AX,[BX]
    0090 89450A        155             MOV       [DI+X_K],AX
    0093 8B5C18        156             MOV       BX,[SI+S_N]          LOAD N
    0096 8B07          157             MOV       AX,[BX]
    0098 89450C        158             MOV       [DI+X_N],AX
    009B 8B4502        159             MOV       AX,[DI+X_CO] .       LOAD C
    009E 89450E        160             MOV       [DI+X_C],AX
    00A1 8B5C08        161             MOV       BX,[SI+S_L]          LOAD L
    00A4 8B07          162             MOV       AX,[BX]
    00A6 894510        163             MOV       [DI+X_L],AX
    00A9 E82601        164             CALL      HACK                 HALT MODE ACK HANDLER
                       165
    00AC E90500        166             JMP       PH2
                       167
    00AF B453          168 SNR         MOV       AH,#SPRO             STORE START OF RECORD
    00B1 E81900        169             CALL      PUTXCHR
                       170
    00B4 E84701        171 PH2         CALL      RDS                  OUTPUT SYNCHRONIZATION


                       172
    00B7 8B5C06        173 CALCJ       MOV       BX,[SI+S_NX]         SI' = SI + 2*Nmax + 1EH
    00BA 8B07          174             MOV       AX,[BX]
    00BC D1E0          175             SAL       AX,1                 MULT BY 2
    00BE 03F0          176             ADD       SI,AX
    00C0 83C61E        177             ADD       SI,#1EH
    00C3 8B3D          178             MOV       DI,[DI]              ORIG DI
    00C5 037D0C        179             ADD       DI,[DI+R_BR]         DI''= DI + BLR + 12H
    00C8 83C712        180             ADD       DI,#R_RB
                       181 .
    00CB FF24          182             JMP WORD PTR[SI]
                       183
                       184
                       185
                       186
                       187 *** THIS ROUTINE PUTS AN ASCII CHARACTER
                       188 *   INTO THE TRANSMITTER BUFFER.  THE
                       189 *** CHAR IS PASSED THROUGH AH.
                       190
    00CD 53            191 PUTXCHR     PUSH      BX
    00CE 8B5D06        192             MOV       BX,[DI+X_IX]         GET IPX
    00D1 3B5D04        193             CMP       BX,[DI+X_BX]         IPX >= BLX ?
    00D4 7C02          194             JL        CSKIP
                       195
    00D6 33DB          196             XOR       BX,BX                YES, RESET IPX
    00D8 8B611A        197 CSKIP       MOV       [BX][DI+X_XB],AH     SEND CHAR
    00DB 43            198             INC       BX
    00DC 895D06        199             MOV       [DI+X_IX],BX         UPDATE IPX
    00DF 5B            200             POP       BX
    00E0 C3            201             RET
                       202
                       203
                       204
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    00E1 51              205 CINIT      PUSH         CX
    00E2 52              206            PUSH         DX
    00E3 8B5C02          207            MOV          BX,[SI+S_BR]
    00E6 8B07            208            MOV          AX,[BX]                    GET BLR
    00E8 89450C          209            MOV          [DI+R_BR],AX               LOAD BLR
    00EB C74508FFFF      210            MOV WORD PTR[DI+R_IT],#0FFFFH           SET INIT FLAG
    00F0 8B5C1A          211            MOV          BX,[SI+S_DV]
    00F3 C7070000        212            MOV WORD PTR[BX],#0000H                 RESET DVRT
    00F7 8B5C0C          213            MOV          BX,[SI+S_XR]
    00FA 8B1F            214            MOV          BX,[BX]                    GET PTX
    00FC 891D            215            MOV          [DI],BX                    LOAD PTX
    00FE 893F            216            MOV          [BX],DI                    LOAD PTR WITH DI
    0100 8B5C06          217            MOV          BX,[SI+S_NX]
    0103 8B0F            218            MOV          CX,[BX]                    GET Nmax
    0105 894D04          219            MOV          [DI+R_NX],CX               LOAD Nmax
    0108 8B5C18          220            MOV          BX,[SI+S_N]                INITIALIZE N=Nmax
    010B 890F            221            MOV          [BX],CX
    010D D1E1            222            SAL          CX,1                       DOUBLE Nmax
    010F 8B5C04          223            MOV          BX,[SI+S_B]
    0112 8B07            224            MOV          AX,[BX]                    GET B
    0114 33D2            225            XOR          DX,DX                      CLR UPPER WORD OF DVND
    0116 F7F9            226            IDIV         CX                         C = B/2#Nmax
    0118 8B3D            227            MOV          DI,[DI]                    POINT TO XMIT BUFFER
    011A 894502          228            MOV          [DI+X_CO],AX               STORE C#


    011D 8B07            229            MOV          AX,[BX]                    GET B
    011F 8B5C08          230            MOV          BX,[SI+S_L]
    0122 8B0F            231            MOV          CX,[BX]                    GET L#
    0124 F7E9            232            IMUL         CX                         B # L#
    0126 052000          233            ADD          AX,#20H                    BLX = B # L# + 8
    0129 894504          234            MOV          [DI+X_BX],AX               LOAD BLX
    012C C745060000      235            MOV WORD PTR[DI+X_IX],#0000H            RESET IPX
    0131 C745080000      236            MOV WORD PTR[DI+X_OX],#0000H            RESET OPX
    0136 C745120000      237            MOV WORD PTR[DI+X_RI],#0000H            RESET RICNT
    013B C745140000      238            MOV WORD PTR[DI+X_RB],#0000H            RESET RBCNT
    0140 C74518FFFF      239            MOV WORD PTR[DI+X_EG],#0FFFFH           SET EORFLG
    0145 5A              240            POP          DX
    0146 59              241            POP          CX
    0147 C3              242            RET
                         243
                         244
                         245
    0148 53              246 DDS        PUSH         BX                         STORE LIST PTR
    0149 8B581C          247            MOV          BX,[BX][SI+S_CH]           GET STORAGE LOC
    014C 8B1F            248            MOV          BX,[BX]                    GET DATA ADDR
    014E 8B07            249            MOV          AX,[BX]                    GET DATA
    0150 E84300          250            CALL         SADJUST                    CNVRT TO 8 BIT
    0153 E81A00          251            CALL         TOASC                      CNVRT TO ASCII
    0156 E874FF          252            CALL         PUTXCHR                    STORE UPPER CHAR
    0159 8AE0            253            MOV          AH,AL
    015B E86FFF          254            CALL         PUTXCHR                    STORE LOWER CHAR
    015E 5B              255            POP          BX
```

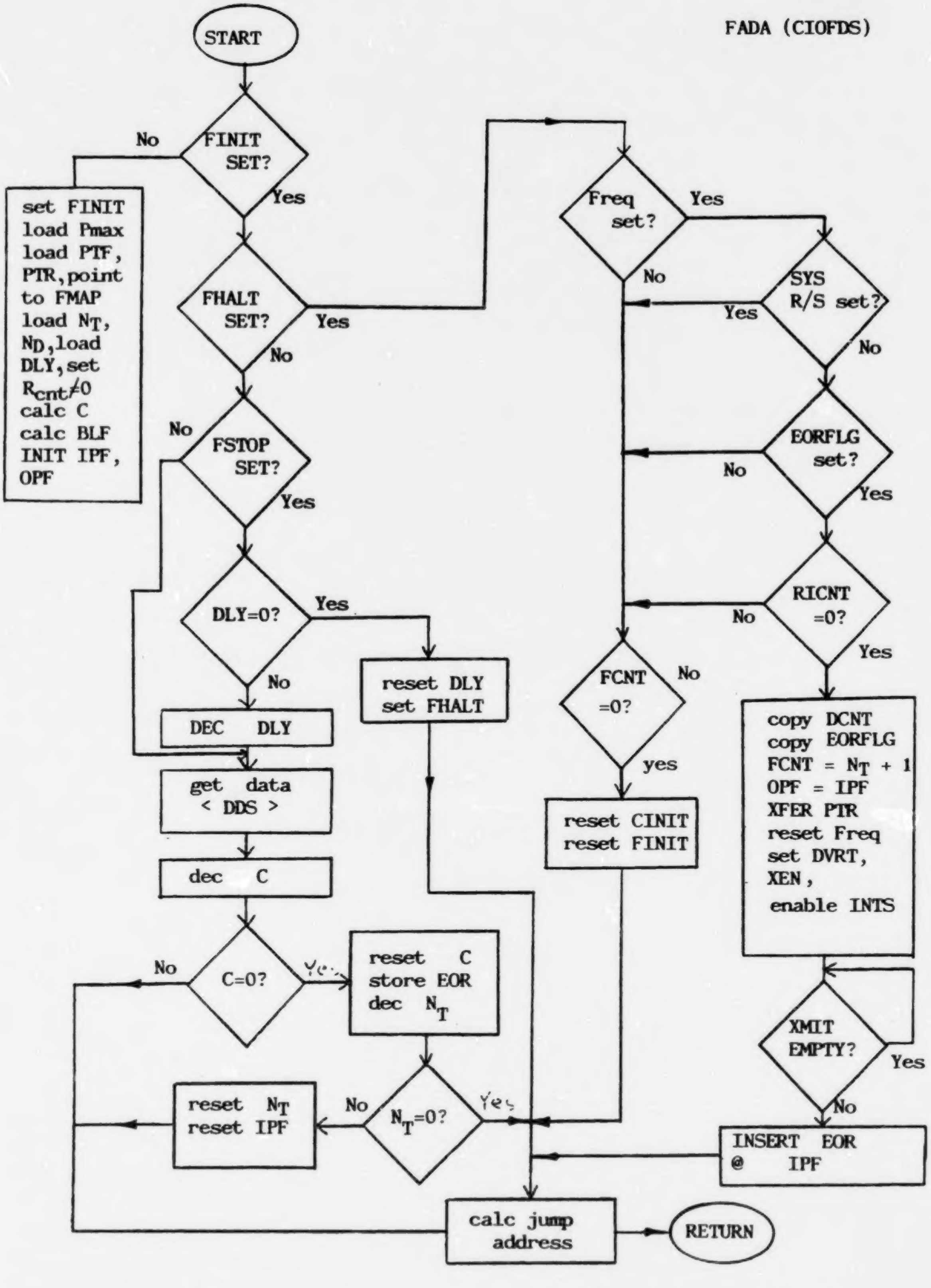
```
LOCATION OBJECT CODE LINE     SOURCE LINE

   015F 83C302        256            ADD        BX,#02H               INC LIST PTR
   0162 FF4D0C        257            DEC WORD PTR[DI+X_N]             DEC N
   0165 75E1          258            JNZ        DDS
                      259
   0167 8B5C18        260            MOV        BX,[SI+S_N]           RESET N
   016A 8B07          261            MOV        AX,[BX]
   016C 89450C        262            MOV        [DI+X_N],AX
   016F C3            263            RET
                      264
                      265
                      266
                      267 ***     THIS ROUTINE TAKES A HEX BYTE NUMBER
                      268 *        AND CONVERTS IT TO TWO ASCII CHARACTERS.
                      269 *       THE HEX BYTE IS PASSED THROUGH AL AND THE
                      270 *        UPPER ASCII AND LOWER ASCII CHARS ARE
                      271 ***      RETURNED THROUGH AH AND AL RESPECTIVELY.
                      272
   0170 8AE0          273 TOASC      MOV        AH,AL
   0172 D0EC          274            SHR        AH,1
   0174 D0EC          275            SHR        AH,1
   0176 D0EC          276            SHR        AH,1
   0178 D0EC          277            SHR        AH,1
   017A 80FC0A        278            CMP        AH,#0AH
   017D 7C06          279            JL         DEC1
   017F 80C437        280            ADD        AH,#37H
   0182 E90300        281            JMP        NEXT
   0185 80C430        282 DEC1       ADD        AH,#30H
   0189 240F          283 NEXT       AND        AL,#0FH
   018A 3C0A          284            CMP        AL,#0AH
   018C 7C05          285            JL         DEC2


   018E 0437          286            ADD        AL,#37H
   0190 E90200        287            JMP        ENDT
   0193 0430          288 DEC2       ADD        AL,#30H
   0195 C3            289 ENDT       RET
                      290
                      291
                      292
                      293 ***      THIS ROUTINE CONVERTS AN 8.0 PU 16 BIT NUMBER
                      294 *        TO A 1.0 PU 8 BIT NUMBER. THE 8 BIT NUMBER
                      295 ***      IS RETURNED IN AL.
                      296
   0196 80FC0F        297 SADJUST    CMP        AH,#0FH
   0199 7614          298            JBE        VLD
   019B 80FCF0        299            CMP        AH,#0F0H
   019E 730F          300            JAE        VLD
   01A0 80FC80        301            CMP        AH,#80H
   01A3 7205          302            JB         OVERPOS
   01A5 B480          303            MOV        AH,#80H
   01A7 E90B00        304            JMP        FINI
   01AA B47F          305 OVERPOS    MOV        AH,#7FH
   01AC E90600        306            JMP        FINI
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

  01AF D1E0         307 VLD       SHL          AX,1
  01B1 D1E0         308           SHL          AX,1
  01B3 D1E0         309           SHL          AX,1
  01B5 8AC4         310 FINI      MOV          AL,AH
  01B7 C3           311           RET
                    312
                    313
                    314
  01B8 8B5C12       315 RACK      MOV          BX,[SI+S_AK]           PENDING ASYNC REQUEST?
  01BB 833F00       316           CMP WORD PTR[BX],#0000H
  01BE 7411         317           JE           RETR                   NO
                    318
  01C0 8A27         319           MOV          AH,[BX]                YES, STORE RECORD
  01C2 E808FF       320           CALL         PUTXCHR
  01C5 B40D         321           MOV          AH,#EOR
  01C7 E803FF       322           CALL         PUTXCHR
  01CA C7070000     323           MOV WORD PTR[BX],#0000H             CLEAR ASYNC REQUEST
  01CE FF4514       324           INC WORD PTR[DI+X_RB]               INC RBCNT
  01D1 C3           325 RETR      RET
                    326
                    327
                    328
  01D2 8B5C12       329 HACK      MOV          BX,[SI+S_AK]           PENDING ASYNC REQUEST?
  01D5 833F00       330           CMP WORD PTR[BX],#0000H
  01D8 7423         331           JE           RETH                   NO
                    332
  01DA 8A27         333           MOV          AH,[BX]                YES, STORE RECORD
  01DC E8EEFE       334           CALL         PUTXCHR
  01DF B40D         335           MOV          AH,#EOR
  01E1 E8E9FE       336           CALL         PUTXCHR
  01E4 C7070000     337           MOV WORD PTR[BX],#0000H             CLEAR ASYNC REQUEST
  01E8 FF4512       338           INC WORD PTR[DI+X_RI]               INC RICNT
  01EB 837D1201     339           CMP WORD PTR[DI+X_RI],#0001H        STILL XMITTING?
  01EF 750C         340           JNE          RETH                   YES
                    341
  01F1 8B5C10       342           MOV          BX,[SI+S_XN]           NO, SET XEN FLAG


  01F4 C707FFFF     343           MOV WORD PTR[BX],#0FFFFH
  01F8 C60602A127   344           MOV BYTE PTR[AUXSTA],#27H           ENABLE XMIT INTS
  01FD C3           345 RETH      RET
                    346
                    347
                    348
  01FE 8B5C0E       349 RDS       MOV          BX,[SI+S_DQ]            Delay REQuested?
  0201 833F00       350           CMP WORD PTR[BX],#0000H
  0204 7428         351           JE           SKIPD1                 NO
                    352
  0206 FF4D16       353           DEC WORD PTR[DI+X_DT]               DEC DCouNT
  0209 752B         354           JNZ          RETD                   NO.
                    355
  020B 8B5C0A       356           MOV          BX,[SI+S_DT]           RESET DCNT
  020E 8B07         357           MOV          AX,[BX]
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    0210 894516        358             MOV         [DI+X_DT],AX
    0213 8B5C0E        359             MOV         BX,[SI+S_DQ]           RESET DREQ
    0216 C7070000      360             MOV WORD PTR[BX],#0000H
    021A FF4D12        361             DEC WORD PTR[DI+X_RI]              DEC RICouNT
    021D 7417          362             JZ          RETD
                       363
    021F 8B5C10        364             MOV         BX,[SI+S_XN]            SET XEN FLAG
    0222 C707FFFF      365             MOV WORD PTR[BX],#0FFFFH
    0226 C60602A127    366             MOV BYTE PTR[AUXSTA],#27H          ENABLE XMIT INTS
    022B E90800        367             JMP         RETD
                       368
    022E 8B5C0A        369 SKIPD1      MOV         BX,[SI+S_DT]            RESET DCNT
    0231 8B07          370             MOV         AX,[BX]
    0233 894516        371             MOV         [DI+X_DT],AX
                       372
    0236 C3            373 RETD        RET

Errors=    0
```

Program:

FADA (CIOFDS)



START
FINIT SET?
No
Yes
set FINIT
load Pmax
load PTF,
PTR, point
to FMAP
load N_T,
N_D, load
DLY, set
$R_{cnt} \neq 0$
calc C
calc BLF
INIT IPF,
OPF
FHALT SET?
Yes
No
FSTOP SET?
No
Yes
DLY=0?
Yes
No
DEC DLY
reset DLY
set FHALT
get data
< DDS >
dec C
C=0?
No
Yes
reset C
store EOR
dec N_T
N_T=0?
No
Yes
reset N_T
reset IPF
calc jump
address
RETURN
Freq set?
Yes
No
SYS R/S set?
Yes
No
EORFLG set?
No
Yes
RICNT =0?
No
Yes
FCNT =0?
No
yes
reset CINIT
reset FINIT
copy DCNT
copy EORFLG
FCNT = N_T + 1
OPF = IPF
XFER PTR
reset Freq
set DVRT,
XEN,
enable INTS
XMIT EMPTY?
Yes
No
INSERT EOR
@ IPF

XMAP

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                      1            "8086"
                      2            NAME  "CIOFDS CALL ADDR-21D2"
                      3
                      4 *          FAULT TRIGGERED DATA STORAGE BLOCK
                      5 *
                      6 *          This block is similar in some ways to
                      7 *          CIOSYS.  Fault data is acquired, processed,
                      8 *          and stored in a separate transmit buffer.
                      9 *          When a fault condition is detected, data
                     10 *          acquisition is halted after a delay sequence.
                     11 *          Upon a download request, CIOSYS is put
                     12 *          in a special mode and the data is transmitted.
                     13
                     14            PROG
                     15
                     16            EXTERNAL    TOASC
                     17            EXTERNAL    SADJUST
                     18
         <A100>      19 AUXIO      EQU         0A100H               AUX SERIAL I/O PORT
         <A102>      20 AUXSTA     EQU         0A102H               AUX SERIAL CMD/STATUS REG
         <0006>      21 ACKPRO     EQU         06H                  ACKNOWLEDGE PROTOCOL
         <0005>      22 ENQPRO     EQU         05H                  FAULT SIGNAL PROTOCOL
         <0053>      23 SPRO       EQU         53H                  "S" DATA START PROTOCOL
         <000D>      24 EOR        EQU         0DH                  END OF RECORD PROTOCOL
                     25
                     26
                     27
                     28 ****  CIOFDS PARAMETER LIST (PASSED VARS AND CONSTANTS)  ****
                     29
         <0000>      30 S_CA       EQU         00H          call addr
         <0002>      31 S_B        EQU         02H          b
         <0004>      32 S_PX       EQU         04H          Pmax
         <0006>      33 S_NT       EQU         06H          NT
         <0008>      34 S_FR       EQU         08H          Faddr*
         <000A>      35 S_XN       EQU         0AH          XEN
         <000C>      36 S_AK       EQU         0CH          ASC
         <000E>      37 S_RS       EQU         0EH          sys R/S flag
         <0010>      38 S_DV       EQU         10H          DVRT flag
         <0012>      39 S_ND       EQU         12H          ND
         <0014>      40 S_DY       EQU         14H          DLY
         <0016>      41 S_FP       EQU         16H          FSTOP
         <0018>      42 S_FT       EQU         18H          FHALT
         <001A>      43 S_FQ       EQU         1AH          FREQ
         <001C>      44 S_CF       EQU         1CH          Fault Channel base pointer
                     45
                     46
                     47
                     48 ****  SYSTEM R MAP (INIT FLAGS/RCVR BUFFER AND POINTERS) ****
                     49
         <0000>      50 R_X        EQU         00H          pointer to X map
         <0002>      51 R_F        EQU         02H          pointer to F map
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

            <0004>     52 R_NX       EQU         04H         Nmax
            <0006>     53 R_PX       EQU         06H         Pmax
            <0008>     54 R_IT       EQU         08H         INIT flag
            <000A>     55 R_FT       EQU         0AH         FINIT flag
            <000C>     56 R_BR       EQU         0CH         BLR
            <000E>     57 R_IR       EQU         0EH         IPR


            <0010>     58 R_OR       EQU         10H         OPR
            <0012>     59 R_RB       EQU         12H         rcvr data base
                       60
                       61
                       62
            <0012>     63 X_RI       EQU         12H         RICNT
                       64
            <0016>     65 X_DT       EQU         16H         DCNT
            <0018>     66 X_EG       EQU         18H         EORFLG
                       67
                       68
                       69
                       70 ****   SYSTEM F MAP (XMTR POINTER AND BUFFER OVERLAY)  ****
                       71
            <0000>     72 F_R        EQU         00H         pointer to R map
            <0002>     73 F_CX       EQU         02H         C#
            <0004>     74 F_BF       EQU         04H         BLF
            <0006>     75 F_IF       EQU         06H         IPF
            <0008>     76 F_OF       EQU         08H         OPF
            <000A>     77 F_NT       EQU         0AH         NT
            <000C>     78 F_ND       EQU         0CH         ND
            <000E>     79 F_C        EQU         0EH         C
            <0010>     80 F_DY       EQU         10H         DLY
            <0012>     81 F_FT       EQU         12H         FCNT (RICNT in X)
            <0014>     82 X_RB       EQU         14H         RBCNT
            <0016>     83 F_DT       EQU         16H         DCNT
            <0018>     84 F_EG       EQU         18H         EORFLG
            <001A>     85 F_FB       EQU         1AH         xmtr data base
                       86
                       87
                       88
  0000 837D0A00        89            CMP WORD PTR[DI+R_FT],#0000H         FINIT SET?
  0004 7506            90            JNE         NOFRST
                       91
  0006 E80A01          92            CALL        FINIT                    NO
  0009 E9F100          93            JMP         CALCJ
                       94
  000C 8B7D02          95 NOFRST     MOV         DI,[DI+R_F]              YES, POINT TO FMAP
  000F 8B5C18          96            MOV         BX,[SI+S_FT]
  0012 833F00          97            CMP WORD PTR[BX],#0000H              HALT SET?
  0015 755A            98            JNE         MODE2                    YES
                       99
  0017 8B5C16         100            MOV         BX,[SI+S_FP]             NO
  001A 833F00         101            CMP WORD PTR[BX],#0000H              STOP SET?
  001D 7409           102            JE          RMODE                    NO
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    103
  001F 837D1000     104              CMP WORD PTR[DI+F_DY],#0000H        YES,DLY = 0?
  0023 7433         105              JE          SETH                    YES
                    106
  0025 FF4D10       107              DEC WORD PTR[DI+F_DY]               NO,DEC DLY
                    108
  0028 33DB         109 RMODE        XOR         BX,BX
  002A E85E01       110              CALL        DDS
                    111
  002D FF4D0E       112              DEC WORD PTR[DI+F_C]                DEC C
  0030 7403         113              JZ          NEWC                    YES
                    114


  0032 E9C800       115              JMP         CALCJ                   NO
                    116
  0035 8B4502       117 NEWC         MOV         AX,[DI+F_CX]            RESET C
  0038 89450E       118              MOV         [DI+F_C],AX
  003B B40D         119              MOV         AH,#EOR                 STORE EOR
  003D E83701       120              CALL        PUTFCHR
                    121
  0040 FF4D0A       122              DEC WORD PTR[DI+F_NT]               DEC NT
  0043 7403         123              JZ          NEWNT                   YES
                    124
  0045 E9B500       125              JMP         CALCJ                   NO
                    126
  0048 8B5C06       127 NEWNT        MOV         BX,[SI+S_NT]            RESET NT
  004B 8B07         128              MOV         AX,[BX]
  004D 89450A       129              MOV         [DI+F_NT],AX
  0050 C745060000   130 ·            MOV WORD PTR[DI+F_IF],#0000H        RESET IPF
  0055 E9A500       131              JMP         CALCJ
                    132
  0058 8B5C14       133 SETH         MOV         BX,[SI+S_DY]            RESET DLY
  005B 8B07         134              MOV         AX,[BX]
  005D 894510       135              MOV         [DI+F_DY],AX
  0060 8B5C18       136              MOV         BX,[SI+S_FT]            SET HALT
  0063 C707FFFF     137              MOV WORD PTR[BX],#0FFFFH
  0067 8B5C0C       138              MOV         BX,[SI+S_AK]            SEND FAULT SIG.
  006A C7070500     139              MOV WORD PTR[BX],#ENQPRO
  006E E98C00       140              JMP         CALCJ
                    141
  0071 8B5C1A       142 MODE2        MOV         BX,[SI+S_FQ]            FREQ SET?
  0074 833F00       143              CMP WORD PTR[BX],#0000H
  0077 746F         144              JE          RSTF1                   NO
                    145
  0079 8B5C0E       146              MOV         BX,[SI+S_RS]            YES, SYS R/S SET?
  007C 833F00       147              CMP WORD PTR[BX],#0000H
  007F 7567         148              JNE         RSTF1                   YES
                    149
  0081 8B3D         150              MOV         DI,[DI]                 NO, POINT TO RMAP
  0083 8B3D         151              MOV         DI,[DI]                 POINT TO XMAP
                    152
  0085 837D1800     153              CMP WORD PTR[DI+X_EG],#0000H        EORFLG SET?
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    00B9 7458         154             JE          RSTF2                      NO
                      155
    00BB 837D1200     156             CMP WORD PTR[DI+X_RI],#0000H           YES, RICNT = 0?
    00BF 7552         157             JNE         RSTF2                      NO
                      158
    0091 8B4516       159             MOV         AX,[DI+X_DT]               YES, GET DCNT
    0094 8B3D         160             MOV         DI,[DI]                    POINT TO RMAP
                      161
    0096 8B5D02       162             MOV         BX,[DI+R_F]                REPLACE X WITH F
    0099 891D         163             MOV         [DI],BX
                      164
    009B 8B7D02       165             MOV         DI,[DI+R_F]                POINT TO FMAP
    009E 894516       166             MOV         [DI+F_DT],AX               LOAD DCNT
    00A1 C74518FFFF   167             MOV WORD PTR[DI+F_EG],#0FFFFH          SET EORFLG
    00A6 8B5C06       168             MOV         BX,[SI+S_NT]
    00A9 8B07         169             MOV         AX,[BX]                    GET NT
    00AB 050100       170             ADD         AX,#1
    00AE 894512       171             MOV         [DI+F_FT],AX               SET UP FCNT


    00B1 8B4506       172             MOV         AX,[DI+F_IF]
    00B4 894508       173             MOV         [DI+F_OF],AX               OPF <- IPF
    00B7 8B5C1A       174             MOV         BX,[SI+S_FQ]
    00BA C7070000     175             MOV WORD PTR[BX],#0000H                RESET FREQ
    00BE 8B5C10       176             MOV         BX,[SI+S_DV]
    00C1 C707FFFF     177             MOV WORD PTR[BX],#0FFFFH               SET DVRT FLAG
    00C5 8B5C0A       178             MOV         BX,[SI+S_XN]
    00C8 C707FFFF     179             MOV WORD PTR[BX],#0FFFFH               SET XEN
    00CC C60602A127   180             MOV BYTE PTR[AUXSTA],#27H              ENABLE XMIT INTS
    00D1 A002A1       181 EMPTY       MOV         AL,[AUXSTA]                WAIT UNTIL
    00D4 2401         182             AND         AL,#01H                    XMIT BUF FULL
    00D6 75F9         183             JNZ         EMPTY
                      184
    00D8 B00D         185             MOV         AL,#EOR
    00DA 8B5D06       186             MOV         BX,[DI+F_IF]               INSERT EOR
    00DD 88411A       187             MOV         [BX][DI+F_FB],AL
    00E0 E91A00       188             JMP         CALCJ
                      189
    00E3 8B3D         190 RSTF2       MOV         DI,[DI]                    POINT TO RMAP
    00E5 8B7D02       191             MOV         DI,[DI+R_F]                POINT TO FMAP
    00E8 837D1200     192 RSTF1       CMP WORD PTR[DI+F_FT],#0000H           FCNT = 0?
    00EC 750F         193             JNE         CALCJ                      NO
                      194
    00EE 8B3D         195             MOV         DI,[DI]                    POINT TO RMAP
    00F0 C745080000   196             MOV WORD PTR[DI+R_IT],#0000H           RESET INIT FLAG
    00F5 C7450A0000   197             MOV WORD PTR[DI+R_FT],#0000H           RESET FINIT FLAG
    00FA 8B7D02       198             MOV         DI,[DI+R_F]                POINT TO FMAP
                      199
    00FD 8B5C04       200 CALCJ       MOV         BX,[SI+S_PX]
    0100 8B07         201             MOV         AX,[BX]
    0102 D1E0         202             SAL         AX,1
    0104 03F0         203             ADD         SI,AX
    0106 83C61E       204             ADD         SI.#1EH
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   0109 8B3D          205            MOV          DI,[DI]                      POINT TO RMAP
   010B 037D0C        206            ADD          DI,[DI+R_BR]                 ADD BLR
   010E 83C712        207            ADD          DI,#R_RB
                      208
   0111 FF24          209            JMP WORD PTR[SI]
                      210
                      211
   0113 C7450AFFFF    212 FINIT      MOV WORD PTR[DI+R_FT],#0FFFFH             SET FINIT
                      213
   0118 8B5C04        214            MOV          BX,[SI+S_PX]                 GET PMAX
   011B 8B07          215            MOV          AX,[BX]
   011D 894506        216            MOV          [DI+R_PX],AX                 LOAD PMAX
   0120 8B5C12        217            MOV          BX,[SI+S_ND]                 INITIALIZE ND=Pmax
   0123 8907          218            MOV          [BX],AX
                      219
   0125 8B5C08        220            MOV          BX,[SI+S_FR]                 GET FADDR
   0128 8B1F          221            MOV          BX,[BX]
   012A 895D02        222            MOV          [DI+R_F],BX                  STORE FADDR
   012D 893F          223            MOV          [BX],DI                      LOAD PTR WITH DI
   012F 8BFB          224            MOV          DI,BX                        POINT TO FMAP
                      225
   0131 8B5C06        226            MOV          BX,[SI+S_NT]                 GET NT
   0134 8B07          227            MOV          AX,[BX]
   0136 89450A        228            MOV          [DI+F_NT],AX                 LOAD NT


                      229
   0139 8B5C12        230            MOV          BX,[SI+S_ND]                 GET ND
   013C 8B07          231            MOV          AX,[BX]
   013E 89450C        232            MOV          [DI+F_ND],AX                 LOAD ND
   0141 8BC8          233            MOV          CX,AX
                      234
   0143 8B5C14        235            MOV          BX,[SI+S_DY]                 GET DLY
   0146 8B07          236            MOV          AX,[BX]
   0148 894510        237            MOV          [DI+F_DY],AX                 LOAD DLY
                      238
   014B C74512FF00    239            MOV WORD PTR[DI+F_FT],#00FFH              SET FCNT <> 0
                      240
   0150 8B5C02        241            MOV          BX,[SI+S_B]                  GET B
   0153 8B07          242            MOV          AX,[BX]
   0155 D1E1          243            SAL          CX,1                         CALC C
   0157 33D2          244            XOR          DX,DX
   0159 F7F9          245            IDIV         CX
   015B 894502        246            MOV          [DI+F_CX],AX
   015E 89450E        247            MOV          [DI+F_C],AX
                      248
   0161 F7E9          249            IMUL         CX                           CALC BLF
   0163 40            250            INC          AX
   0164 F76D0A        251            IMUL WORD PTR[DI+F_NT]
   0167 894504        252            MOV          [DI+F_BF],AX
                      253
                      254
   016A C745060000    255            MOV WORD PTR[DI+F_IF],#0000H              INIT IPF
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

    004A 3C43            103 T1CMD4   CMP      AL,#43H              = 'C'
    004C 7503            104          JNE      EXIT1
                         105
    004E E92501          106          JMP      CMD5
                         107
    0051 E97C01          108 EXIT1    JMP      ENDBLK
                         109
                         110
                         111
                         112
                         113 ***      An 8 bit number specifying the number
                         114 *        of channels in the data set is obtained.


                         115 *        Command N references data base 1.
                         116 ***      Command P references data base 2.
                         117
    0054 E8B701          118 CMD1     CALL     GETBCHR
    0057 3C20            119          CMP      AL,#SPACE
    0059 753C            120          JNE      EXIT2
                         121
    005B E8B001          122          CALL     GETBCHR              UPPER ASCII CHAR.
    005E 8AE0            123          MOV      AH,AL
    0060 E8AB01          124          CALL     GETBCHR              LOWER ASCII CHAR.
    0063 50              125          PUSH     AX
    0064 E8A701          126          CALL     GETBCHR
    0067 3C03            127          CMP      AL,#ETX              PROTOCOL COMPLETED ?
    0069 58              128          POP      AX
    006A 752B            129          JNE      EXIT2
                         130 -
    006C E87B01          131          CALL     TOHEX                FORM SINGLE HEX BYTE
    006F 32E4            132          XOR      AH,AH                CLEAR UPPER BYTE
    0071 80F94E          133          CMP      CL,#4EH              = 'N'
    0074 750B            134          JNE      FBN
                         135
    0076 3B4504          136          CMP      AX,[DI+R_NX]         DATA BASE 1
    0079 7F1C            137          JG       EXIT2                n <= Nmax ?
                         138
    007B 8B5C08          139          MOV      BX,[SI+S_N]          STORE n
    007E E90D00          140          JMP      ADDRN
                         141
    0081 3B4506          142 FBN      CMP      AX,[DI+R_PX]         DATA BASE 2 ('P')
    0084 7F11            143          JG       EXIT2                ND <= Pmax ?
                         144
    0086 C7450A0000      145          MOV WORD PTR[DI+R_FT],#0000H  RESET FINIT FLAG
    008B 8B5C0A          146          MOV      BX,[SI+S_ND]
    008E 8907            147 ADDRN    MOV      [BX],AX              STORE ND
                         148
    0090 B006            149          MOV      AL,#ACK
    0092 8B5C02          150          MOV      BX,[SI+S_AK]
    0095 8807            151          MOV      [BX],AL          ACKNOWLEDGE COMPLETED
                         152                                       ;COMMAND
    0097 E93601          153 EXIT2    JMP      ENDBLK
```

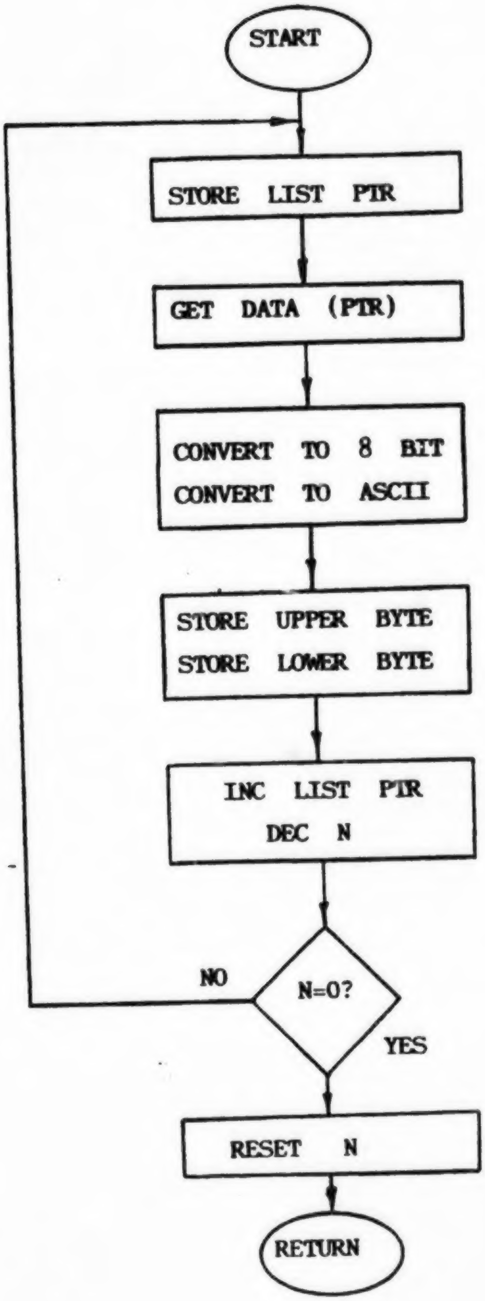
```
LOCATION OBJECT CODE LINE     SOURCE LINE

                    154
                    155
                    156
                    157 ***      This command supplies:
                    158 *                  1)  an 8 bit channel ID no.
                    159 *                  2)  a 16 bit channel address.
                    160 *        'A' references data base 1.
                    161 ***      'E' references data base 2.
                    162
   009A E87101      163 CMD2     CALL     GETBCHR
   009D 3C20        164          CMP      AL,#SPACE
   009F 7568        165          JNE      EXIT3
                    166
   00A1 E86A01      167          CALL     GETBCHR            UPPER ASCII CHAR.
   00A4 8AE0        168          MOV      AH,AL
   00A6 E86501      169          CALL     GETBCHR            LOWER ASCII CHAR.
   00A9 50          170          PUSH     AX
   00AA E86101      171          CALL     GETBCHR


   00AD 3C20        172          CMP      AL,#SPACE
   00AF 58          173          POP      AX
   00B0 7557        174          JNE      EXIT3
                    175
   00B2 E83501      176          CALL     TOHEX              FORM SINGLE HEX BYTE
   00B5 32E4        177          XOR      AH,AH
   00B7 80F941      178          CMP      CL,#41H            = 'A'
   00BA 750E        179          JNE      FBA
                    180
   00BC 8B5C08      181 -        MOV      BX,[SI+S_N]        DATA BASE 1
   00BF 3B07        182          CMP      AX,[BX]            TEST RANGE OF CH. NO.
   00C1 7F46        183          JG       EXIT3
                    184
   00C3 8BD8        185          MOV      BX,AX              CALCULATE OFFSET
   00C5 D1E3        186          SAL      BX,1
   00C7 E91000      187          JMP      ADDRA
                    188
   00CA 8B5C0A      189 FBA      MOV      BX,[SI+S_ND]       DATA BASE 2
   00CD 3B07        190          CMP      AX,[BX]            TEST RANGE OF CH. NO.
   00CF 7F38        191          JG       EXIT3
                    192
   00D1 8B5D04      193          MOV      BX,[DI+R_NX]        CALCULATE OFFSET
   00D4 D1E3        194          SAL      BX,1
   00D6 D1E0        195          SAL      AX,1
   00D8 03D8        196          ADD      BX,AX
                    197
   00DA 8B5812      198 ADDRA    MOV      BX,[BX][SI+S_FQ]   GET STORAGE ADDR
   00DD E82E01      199          CALL     GETBCHR            UPPER ASCII CHAR, 1ST BYTE
   00E0 8AE0        200          MOV      AH,AL
   00E2 E82901      201          CALL     GETBCHR            LOWER ASCII CHAR, 1ST BYTE
   00E5 E80201      202          CALL     TOHEX              FORM SINGLE UPPER BYTE
   00E8 8AF0        203          MOV      DH,AL
   00EA E82101      204          CALL     GETBCHR            UPPER ASCII CHAR, 2ND BYTE
```

```
LOCATION OBJECT CODE LINE     SOURCE LINE

   016F C745080000    256            MOV WORD PTR[DI+F_OF],#0000H        INIT OPF
                      257
   0174 8B3D          258            MOV        DI,[DI]                  POINT TO RMAP
                      259
   0176 C3            260            RET
                      261
                      262
                      263
                      264 ### THIS ROUTINE PUTS AN ASCII CHARACTER
                      265 #   INTO THE FAULT DATA BUFFER.  THE
                      266 ### CHAR IS PASSED THROUGH AH.
                      267
   0177 53            268 PUTFCHR    PUSH       BX
   0178 8B5D06        269            MOV        BX,[DI+F_IF]             GET IPX
   017B 3B5D04        270            CMP        BX,[DI+F_BF]             IPX >= BLX ?
   017E 7C02          271            JL         CSKIP
                      272
   0180 33DB          273            XOR        BX,BX                    YES, RESET IPX
   0182 88611A        274 CSKIP      MOV        [BX][DI+F_FB],AH         SEND CHAR
   0185 43            275            INC        BX
   0186 895D06        276            MOV        [DI+F_IF],BX             UPDATE IPX
   0189 5B            277            POP        BX
   018A C3            278            RET
                      279
                      280
                      281
   018B 53            282 DDS        PUSH       BX                       STORE LIST PTR
   018C 8B581C        283            MOV        BX,[BX][SI+S_CF]         GET STRGE LOC.
   018F 8B1F          284            MOV        BX,[BX]                  GET DATA ADDR
   0191 8B07          285 -          MOV        AX,[BX]                  GET DATA


   0193 E80000        286            CALL       SADJUST                  CNVRT TO 8 BIT
   0196 E80000        287            CALL       TOASC                    CNVRT TO ASCII
   0199 E8DBFF        288            CALL       PUTFCHR                  STORE UPPER CHAR
   019C 8AE0          289            MOV        AH,AL
   019E E8D6FF        290            CALL       PUTFCHR                  STORE LOWER CHAR
   01A1 5B            291            POP        BX
   01A2 83C302        292            ADD        BX,#02H                  INC LIST PTR
   01A5 FF4D0C        293            DEC WORD PTR[DI+F_ND]               DEC ND
   01A8 75E1          294            JNZ        DDS
                      295
   01AA 8B5C12        296            MOV        BX,[SI+S_ND]             RESET ND
   01AD 8B07          297            MOV        AX,[BX]
   01AF 89450C        298            MOV        [DI+F_ND],AX
   01B2 C3            299            RET

Errors=    0
```

Subroutines for:

CODA (CIOSYS)

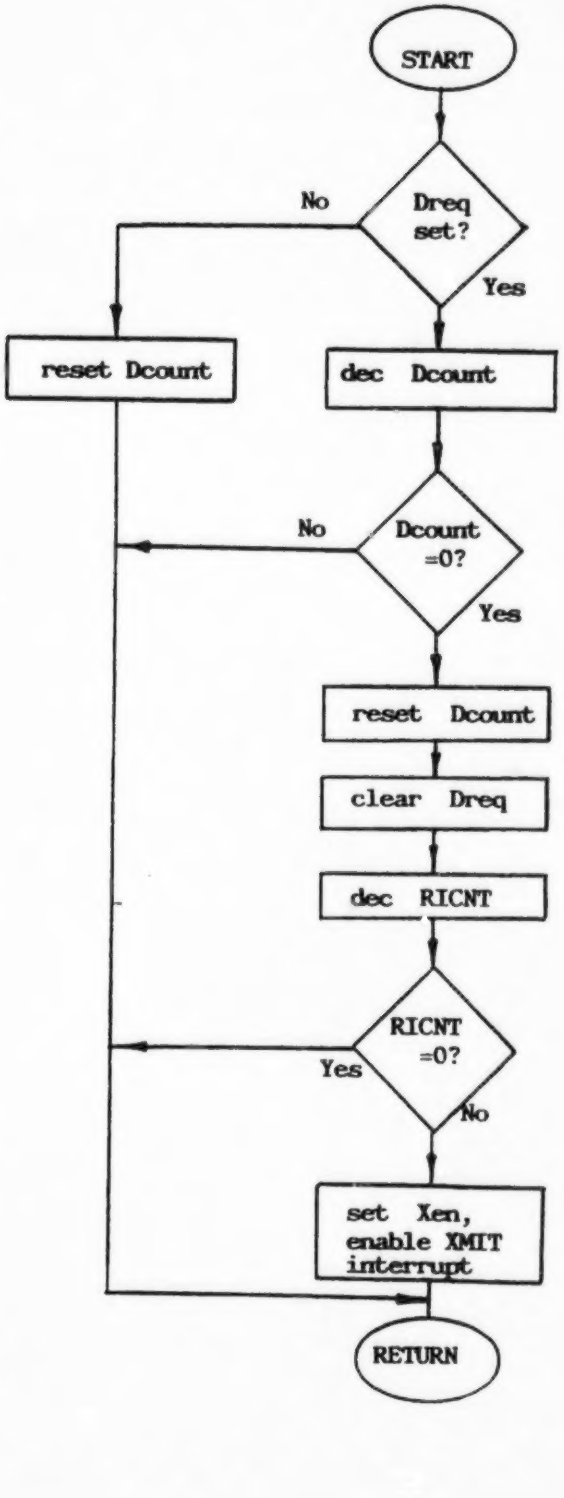
FADA (CIOFDS)


START
STORE LIST PTR
GET DATA (PTR)
CONVERT TO 8 BIT
CONVERT TO ASCII
STORE UPPER BYTE
STORE LOWER BYTE
INC LIST PTR
DEC N
NO
N=0?
YES
RESET N
RETURN

Subroutines for:

CODA (CIOSYS)

FADA (CIOFDS)


START
Dreq set?
No
Yes
reset Dcount
dec Dcount
Dcount =0?
No
Yes
reset Dcount
clear Dreq
dec RICNT
RICNT =0?
Yes
No
set Xen, enable XMIT interrupt
RETURN

MCOS3 BLOCK SPECIFICATIONS

The purpose here is to give a "users" description of the blocks that make up the MCOS3 communication system. While there are certain conceptual differences, these descriptions should resemble those given on pages 60-78 in Section 8 of the BBC blue binder. While actual addresses are specified below, these are only a documentation of the actual system that has been implemented; like the regular MICAS blocks, the user is free to choose his own working address nodes. Finally, the names given for addresses are the names used in the source assembly programs and all other documentation.

1. CIOINIT : 21C6 MCOS3 INITIALIZATION
   EXECUTION TYPE : INITIALIZATION BLOCK

1 F***
2 21C6

2. CIOINT_Y5 : 21C9 I/O INTERRUPT BLOCk
   EXECUTION TYPE : HIGHEST PRIORITY EVENT TRIGGERED BLOCK

1 F***		
2 21C9		
3 Dreq	var out (SYS)	E900
4 XEN	var in/out (SYS/FDS)	E902

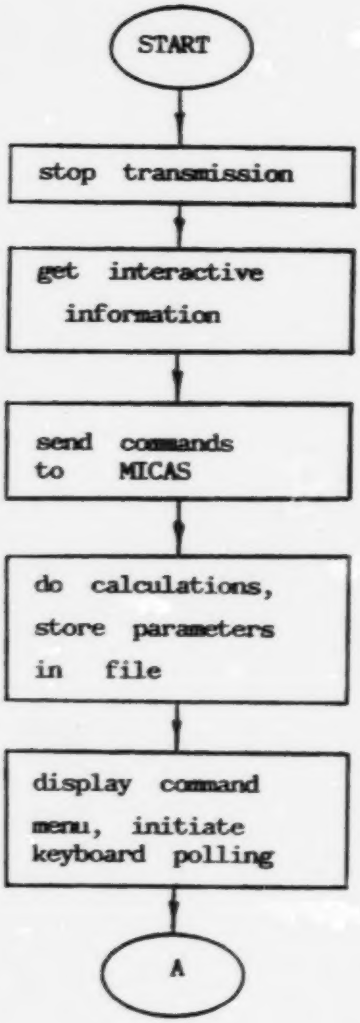
3. CIOBGR : 21CC BACKGROUND COMMAND INTERPRETER
   EXECUTION TYPE : LOW PRIORTIY CYCLICAL BLOCK

1 F***		
2 21CC		
3 ASC	var out (SYS)	E904
4 SYS R/S Flag	var out (SYS/FDS)	E906
5 K	var out (SYS)	E908
6 n	var out (SYS)	E90A
7 N_D (p)	var out (FDS)	E930
8 DLY	var out (FDS)	E932
9 FSTOP	var out (FDS)	E934
10 FHALT	var out (FDS)	E936
11 Freq	var out (FDS)	E938
12 Ch1 ptr	continuous	E90E
	data $n = n_{max}$*	E910
Chn ptr	channels	
ChF1 ptr	fault	
	data $p = p_{max}$*	
ChFP ptr	channels	

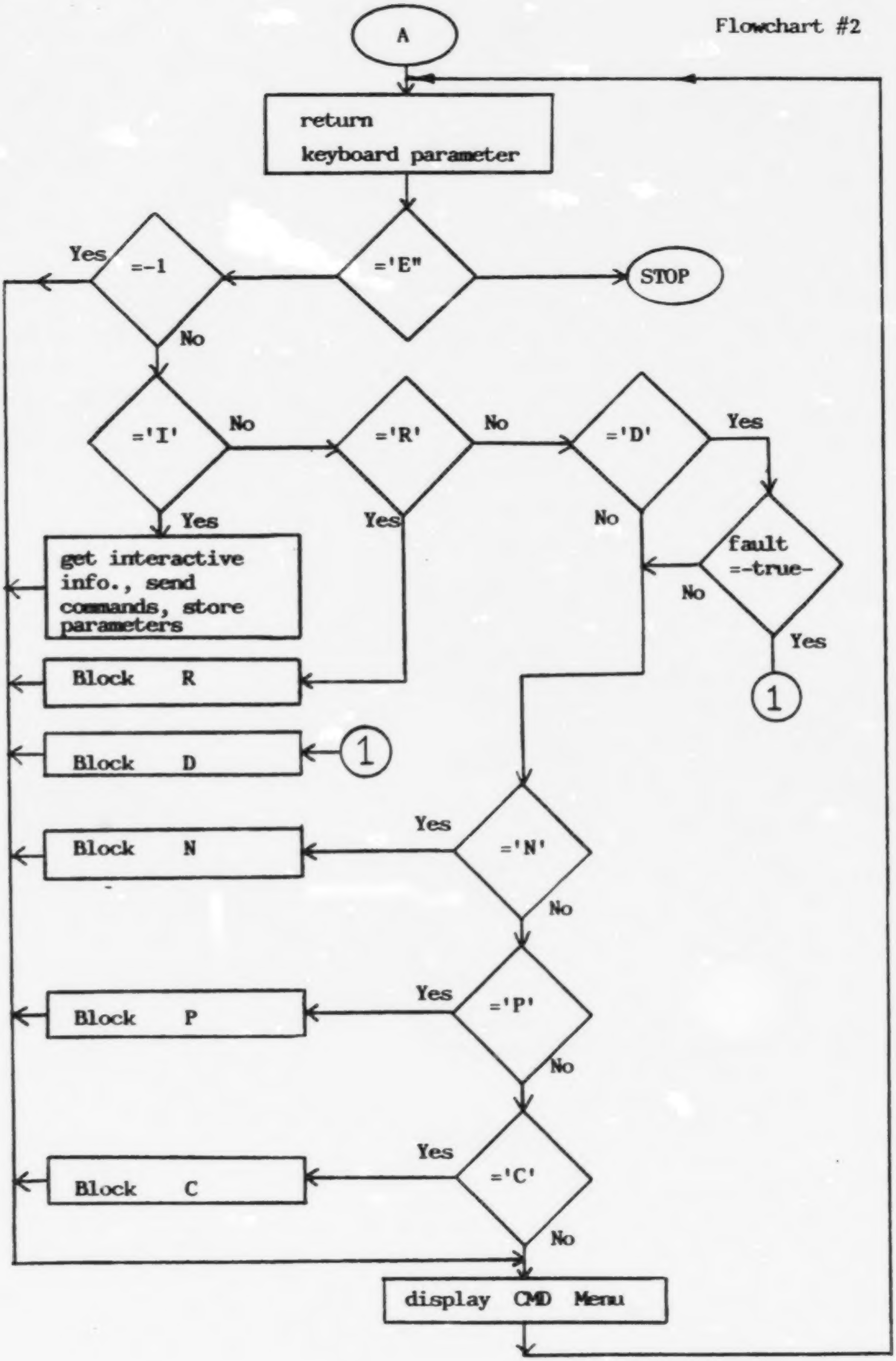
*Note:

n_{max} and p_{max} are constants which are specified while programming CIOSYS and CIOFDS respectively. The number of address spaces allocated for

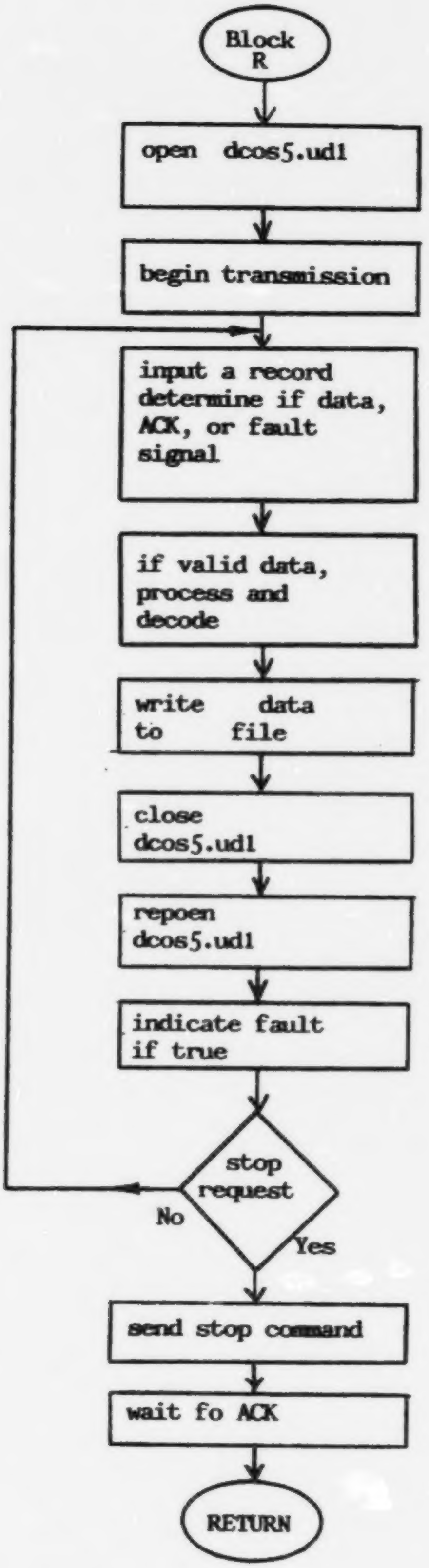
Flowchart #1


START
stop transmission
get interactive information
send commands to MICAS
do calculations, store parameters in file
display command menu, initiate keyboard polling
A

Flowchart #2

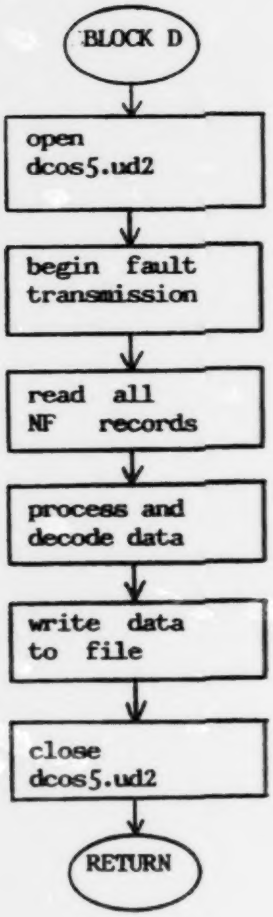

A
return
keyboard parameter
='E"
STOP
=-1
Yes
No
='I'
No
Yes
get interactive info., send commands, store parameters
='R'
No
Yes
Block R
='D'
Yes
No
fault =-true-
No
Yes
1
Block D
1
='N'
Yes
No
Block N
='P'
Yes
No
Block P
='C'
Yes
No
Block C
display CMD Menu

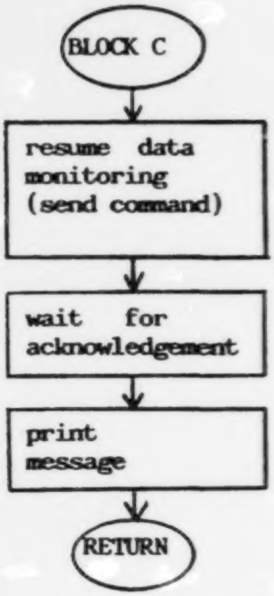
Flowchart #3


Block R
open dcos5.udl
begin transmission
input a record determine if data, ACK, or fault signal
if valid data, process and decode
write data to file
close dcos5.udl
repoen dcos5.udl
indicate fault if true
stop request
No
Yes
send stop command
wait fo ACK
RETURN

DCOS5

Flowchart #4

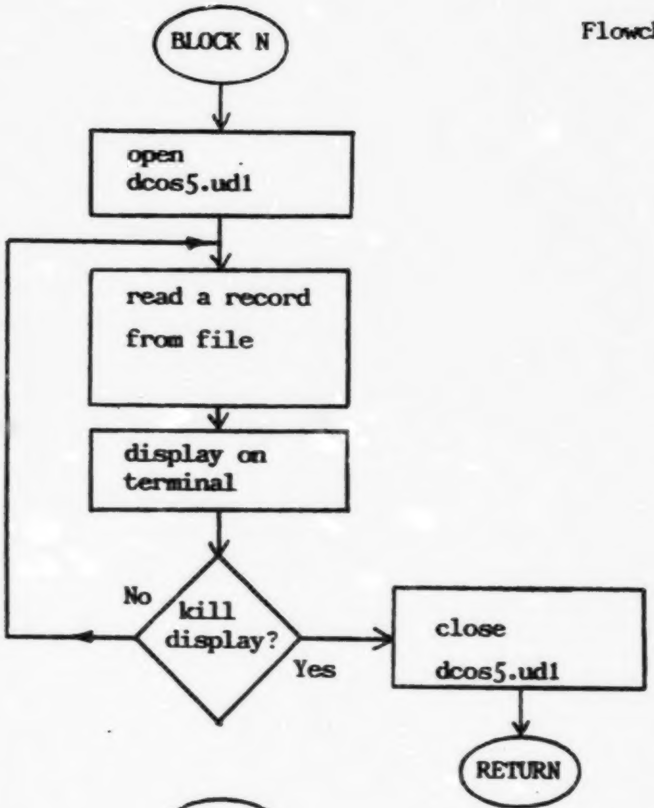

BLOCK D
open dcos5.ud2
begin fault transmission
read all NF records
process and decode data
write data to file
close dcos5.ud2
RETURN


BLOCK C
resume data monitoring (send command)
wait for acknowledgement
print message
RETURN

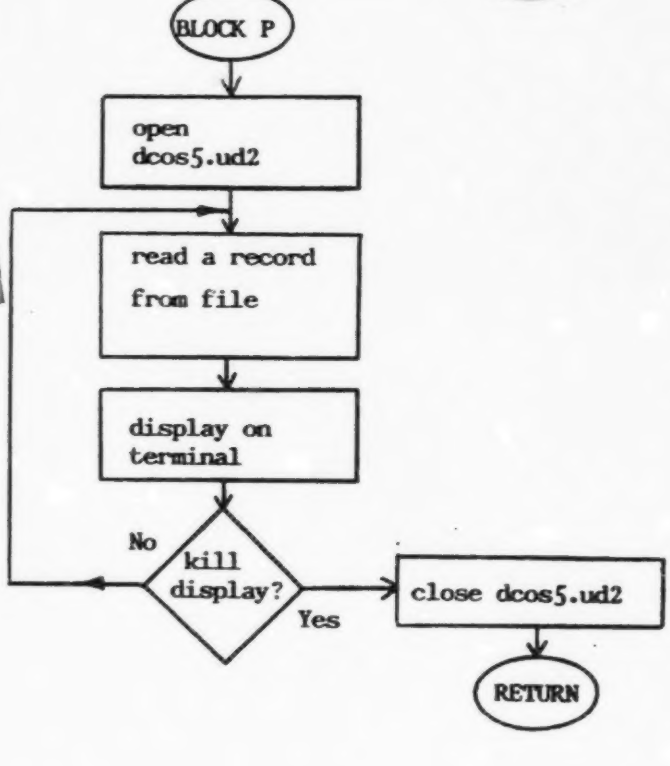
CA1
DTC800
87
S52
c.1.

DCOS5

Flowchart #5


BLOCK N
open dcos5.ud1
read a record from file
display on terminal
No
kill display?
Yes
close dcos5.ud1
RETURN

Library and Information Centre / Centre de Documentation (CSLA)
RECEIVED - REÇU LE
1987
Transports Canada / Transports Canada


BLOCK P
open dcos5.ud2
read a record from file
display on terminal
No
kill display?
Yes
close dcos5.ud2
RETURN

356324